ROBÔS DOMÉSTICOS

Da ficção, os robôs já chegaram às lojas. Mas como ter certeza de que se está realmente comprando um? Neste artigo, você tem as respostas para não levar brinquedo por robô. Ou gato por lebre.

Das páginas dos livros, onde surgiram pela primeira vez, os robôs invadiram revistas, jornais e até mesmo filmes. Nas últimas décadas, eles se materializaram nos laboratórios, foram aproveitados na indústria e, hoje, tornaram-se tão triviais que se encontram à venda em lojas, nos países mais desenvolvidos.

O número cada vez maior de pessoas interessadas pela robótica foi o principal responsável pela difusão dos robôs, vendidos prontos ou em forma de kits para montagem. O maior campo de aplicações dessas máquinas encontra-se na educação: para entender os princípios da robótica, nada melhor do que montar robôs e depois programá-los para tarefas úteis (como a soldagem de componentes eletrônicos numa placa de circuito impresso) ou complexas (por exemplo, servir uma xícara de café).

Os robôs que conseguem fazer isso, no entanto, são muito caros: além do hardware eletrônico, exigem componentes mecânicos que incluem eixos, garras especiais, sensores, motores, circuitos hidráulicos e outras peças — sem as quais não seriam máquinas tão versáteis.

Todavia, quem estiver interessado em compreender o funcionamento dos robôs, não precisa, no início, adquirir um modelo caro e de muitos recursos. Os mais simples, como os mostrados na página seguinte, servem perfeitamente para essa finalidade. Não é incomum que a indústria de brinquedos, no mundo inteiro, anuncie alguns de seus produtos como sendo robôs. Assim, deve-se ter cuidado para não confundi-los com os verdadeiros robôs. Mas como saber as diferenças?

Nem tudo o que se move é robô

Um dos critérios mais importantes se refere ao movimento. Não se espera que um robô programe a si mesmo, ou que escolha uma seqüência

Descubra o robô

Robô
Controlado por um computador externo, este veículo dispõe de sensores de luz e de pressão.

Big Trak
Segue instruções sobre direção e distância semelhantes às do LOGO. Introduzidas pelo teclado, são programadas em seu próprio microprocessador.

Bumper Car
Equipado com motor a pilha, só anda para a frente. Ao bater num obstáculo vira 90° à direita e continua.

Quem sai dessa?
O teste do labirinto pode ser definitivo para determinar qual dos três dispositivos é um robô. O Bumper Car vira à direita sempre que bate em alguma parede — e pára quando há uma exatamente à sua direita. O Big Trak percorrerá o labirinto com exatidão se as instruções do operador estiverem corretas. Por fim, o robô, com seus sensores, é o único capaz de aprender o traçado, qualquer que ele seja, até sair do labirinto.
Agora, imagine o seguinte: o Bumper Car bate no Big Trak e o desvia 90° de seu curso. Com o choque, o Bumper Car também vira automaticamente à direita; e o Big Trak, sem condições de saber que o curso foi alterado, continua como se nada houvesse acontecido. Só o robô teria uma reação "inteligente", tratando a mudança de rumo como apenas um dos aspectos do ambiente imprevisível pelo qual ele está circulando.

de ações sem orientação humana; mas, uma vez colocado em movimento, ele deverá ser capaz de operar sem necessidade de um controle contínuo. Sem essa liberdade de movimento, um aparelho não é um robô.

Outra avaliação diz respeito ao modo de produção do movimento. Um carrinho de brinquedo com motor a pilhas anda sempre em linha reta. Instale um pára-choques, sensor, mais um dispositivo para virar as rodas dianteiras, e ele se desviará de obstáculos como paredes e móveis. Mude seu centro de gravidade, equipe-o com grandes pneus de borracha e ele poderá subir uma rampa muito inclinada, virar e continuar andando. Embora realize tudo isso sem qualquer controle humano, ninguém vai considerá-lo um robô.

Classifica-se o movimento dos robôs em duas categorias: controlado por um programa e com movimento inteligente. Não se pode programar o carrinho que imaginamos para que se movimente de outro modo; portanto não é um robô. Existem, contudo, brinquedos desse tipo programados por meio de cartões perfurados. Dispositivos mecânicos lêem os cartões e reproduzem os movimentos aí codificados. Nesse caso, estamos diante de um robô.

Ainda há outros aspectos a levar em conta. Por exemplo, a máquina em questão tem sensores para receber informações visuais ou auditivas do mundo exterior? Ela reage a estímulos externos modificando alguma de suas características? Ou — passando para um plano de maior complexidade — ela consegue jogar xadrez? Produz respostas sonoras, simulando a fala humana? A característica essencial, no entanto, para se distinguir entre robôs e brinquedos é o movimento programado.

NOME: Beasty
TIPO: Braço e garra
PROGRAMÁVEL? Sim
SOFTWARE FORNECIDO? Sim
SENSORES: De posição

Robô comandado pelo BBC Micro, do qual recebe energia elétrica e sinais de controle por meio do conector de expansões. Utiliza três servomotores. A documentação compõe-se de dois manuais: um referente à montagem do aparelho e outro à sua operação. Ele tem seu próprio sistema operacional, o Robol, por meio do qual se faz a programação dos movimentos (pode-se acionar um motor de cada vez).

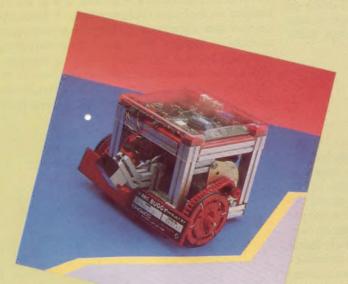
NOME: Hebot II
TIPO: Robô de superfície
PROGRAMÁVEL? Sim
SOFTWARE FORNECIDO? Sim
SENSORES: De toque

O Hebot II é um dos tipos mais populares de robôs destinados à educação: projetado para uso com micros da família Sinclair, tem quatro sensores para evitar colisões, um suporte retrátil para caneta (usado em desenhos) e dois motores de corrente contínua, um para cada roda. Segundo o fabricante, embora projetado para o Sinclair, pode ser adaptado para qualquer micro.

NOME: Memocon Crawler
TIPO: Robô de superfície
PROGRAMÁVEL? Sim
SOFTWARE FORNECIDO? Sim
SENSORES: Não

Controlar um robô completamente "cego", como este, pode ser um grande desafio: o Memocon vem equipado com um conjunto de cinco teclas, para que luzes, campainhas e comandos sejam acionados e combinados. Usa dois motores alimentados por pilhas.

NOME: Valiant Turtle
TIPO: Robô de superfície
PROGRAMÁVEL? Sim
SOFTWARE FORNECIDO? Sim
SENSORES: De posição

Robô bastante sofisticado, tem forma de tartaruga, dois motores (um para cada roda), suporte retrátil para caneta e baterias recarregáveis. Comunica-se com o micro por meio de raios infravermelhos. O software que acompanha a máquina foi desenvolvido na linguagem LOGO, mas isso não impede que funcione com outra linguagem de comando.

NOME: BBC Buggy
TIPO: Robô de superfície
PROGRAMÁVEL? Sim
SOFTWARE FORNECIDO? Sim
SENSORES: De luz e de pressão

Projetado para uso nas escolas inglesas, liga-se ao BBC Micro. Vendido em kits, tem dois motores, alimentação fornecida pela fonte auxiliar do computador, suporte para caneta e, opcionalmente, dispositivo para leitura de código de barras. Com seus sensores, desvia-se de obstáculos e localiza uma fonte de luz.

NOME: Edinburgh Turtle
TIPO: Robô de superfície
PROGRAMÁVEL? Sim
SOFTWARE FORNECIDO? Sim
SENSORES: De posição

Edinburgh é o nome da cidade escocesa onde se desenvolveu esse robô. Ele se liga ao micro por meio de um cabo flexível, pelo qual recebe a energia para acionar seus dois motores (um para cada roda). Os acessórios incluem um suporte retrátil para caneta (o robô também funciona como plotter), um alto-falante e, opcionalmente, um módulo para controle remoto.

ESTRUTURA VERSÁTIL

O uso de funções no interior dos parâmetros evidencia a integridade estrutural do PASCAL. Veremos aqui como se cria um algoritmo "informal" — recurso útil na organização de arquivos.

Assim como definimos e chamamos procedimentos por nomes, as funções também podem ser chamadas, mas apenas em expressões — nunca como instruções. As chamadas de procedimentos implicam execução de subprogramas responsáveis por algum processo, ao passo que as de funções somente *calculam* valores. Estes podem ser de qualquer tipo simples, real ou escalar, ou um tipo "indicador" (que examinaremos mais adiante). A única diferença no cabeçalho, além do uso da palavra reservada FUNCTION, está na necessidade de especificarmos o identificador de tipo do resultado. Supondo que o PASCAL não predefinisse a função ODD, poderíamos facilmente defini-la nós mesmos:

```
FUNCTION ODD (NUMERO :
  INTEGER) : BOOLEAN;
  BEGIN
    ODD := NUMERO MOD 2 > 0
  END; {ODD}
```

O valor é retornado no identificador da função, devendo então ser atribuído a ele no corpo da função, como mostramos. Nesse sentido, os nomes das funções equivalem a variáveis nunca inicializadas, e seus valores são calculados sempre que surgem numa expressão. No corpo da função, eles aparecem apenas no LHS (Left Hand Side, "lado esquerdo") da atribuição; se formos descuidados (e prolixos), programando a atribuição como

```
IF NUMERO MOD 2 > 0 THEN
  ODD := TRUE
```

(esquecendo a sentença ELSE ODD := FALSE), poderá haver uma chamada de função que produzirá um resultado indefinido.

Tal como acontece com os parâmetros de procedimentos já examinados, transfere-se o valor do parâmetro efetivo para o identificador integer local (NUMERO) a partir do ponto de "execução". Assim,

```
WRITELN (ODD (SQR (N DIV 100)))
```

que apresenta FALSE como resposta, executa quatro operações:

1) calcula a expressão N DIV 100;
2) passa esse valor integer temporário para SQR como um parâmetro efetivo de valor;
3) fornece seu quadrado e passa-o para ODD; e
4) calcula o resultado booleano e transfere-o como parâmetro para o procedimento WRITELN.

Se N tiver valor 17 enquanto se executa a instrução, qual será seu valor depois desta? Essa pergunta deve parecer ridícula, pois não há razão para se alterar o valor de N durante o cálculo de uma função que o inclua como parâmetro. Os resultados das funções dependem apenas do "mundo como ele é" — ou seja, o valor obtido não passa de uma "função" (sic) do(s) valor(es) de seus "argumentos" ou parâmetros. No PASCAL, o mecanismo para transferir apenas o valor das variáveis garante que, mesmo com a modificação local dos parâmetros no corpo da função (por serem cópias locais), os parâmetros efetivos não serão alterados pela chamada.

Dados globais x parâmetros

Por essa mesma razão, embora o PASCAL não proíba, os dados devem ser acessados globalmente. Toda vinculação de dados entre procedimentos e chamadas de funções é controlada por meio de listas de parâmetros — mesmo que os dados estejam no âmbito do subprograma. A única exceção a essa regra geral diz respeito ao acesso de constantes globais. Por sua própria estrutura, elas não correm qualquer perigo no PASCAL, pois é impossível alterar seus valores. No entanto, quando se fornece um valor de constante como parâmetro, ele se tornará uma variável local e, portanto, não confiável.

```
FUNCTION MINUSCULA
  (CARACTERE : CHAR)          : CHAR;
  {RETORNA A MINUSCULA P/ QUALQUER
    MAIUSCULA}
  CONST
      DISTANCIA = 32; {ASCII DE ORD
        ('a') - ORD ('A')}
  BEGIN
  IF CARACTERE IN ['A' .. 'Z']
    THEN
      MINUSCULA := CHR (ORD
        (CARACTERE) + DISTANCIA)
    ELSE
      MINUSCULA := CARACTERE
  END; {MINUSCULA}
```

Ao definirmos procedimentos, por vezes torna-se essencial alterar os valores de seus parâmetros — caso contrário, não executarão a tarefa para a qual foram projetados. Um exemplo é o próprio procedimento READ. De nada serviria que READ (N) desse um valor a um número inteiro local a READ e o valor de N permanecesse inalterado. Em tais casos, precisamos transferir o endereço de um parâmetro de variável (em vez de seu valor) para que o procedimento remeta diretamente a ele (e não a uma cópia local). Esse mecanismo recebe o nome de "passagem por

endereço" ou "por referência" e, de modo geral, deve ser usado apenas com procedimentos.

Obtém-se o fornecimento de parâmetros de variáveis pela inclusão da palavra reservada VAR antes de qualquer item (ou itens) na lista de parâmetros do procedimento. A sintaxe de uma lista de parâmetros assemelha-se à declaração VAR de um bloco; mas, ao invés de VAR aparecer como delimitador, emprega-se a declaração antes de um item a ser VARiado por um procedimento. Por exemplo,

```
PROCEDURE PROCESSO (VAR
  CONTADOR : LISTACONT);
```

Desenvolvimento de aplicações

Como ilustração dessa poderosa técnica, criaremos um programa capaz de ler alguns nomes introduzidos por meio do teclado e, vinculada a cada nome, uma quantia em dinheiro devida pela pessoa. Para simplificar, empregaremos apenas um string por nome. Uma vez testado, o programa poderia abranger outros detalhes, como endereços, números telefônicos etc. ou, ainda, imprimir os nomes em ordem alfabética ou segundo as quantias devidas.

A estrutura natural dos dados implica uma lista de registros com campos de tipo apropriado, que depois poderão ser incluídos com facilidade. Podemos lidar com cada registro como se fosse uma unidade isolada de dados, mas usando qualquer dos campos que escolhermos como uma "chave" para a ordenação. Em primeiro lugar, vamos determinar a representação completa dos dados:

```
CONST
  TAMSTRING     = 20;
  TAMLISTA      = 50;
TYPE
  CARDINAL      = 0 .. MAXINT;
  COMPSTRING    = 1 .. TAMSTRING;
  STRING        = PACKED ARRAY
                    [COMPSTRING] OF CHAR;
  DADOS         = RECORD
                    NOME : STRING;
                    DEBITO : CARDINAL
                  END; {DADOS}
  LIMITES       = 1 .. TAMLISTA;
  REGISTROS     = ARRAY [LIMITES]
                    OF DADOS;
```

Essa especificação opera com cinqüenta nomes e supõe (por enquanto) que nenhum excederá vinte caracteres. Contudo, por meio de definições, é possível modificar esses limites.

Todas as subranges e estruturas foram expressas numa definição TYPE por vários motivos:

1) segurança — por exemplo, um índice de tipo LIMITES nunca excede os limites da matriz;
2) depuração de erros — se ocorrer um "range error" durante o teste, a mensagem de erro ajudará a localizar o problema;
3) conveniência e eficácia — as variáveis locais têm tipos existentes, evitando duplicação e economizando memória;
4) necessidade — o PASCAL exige que "indica-

Vantagem positiva

O programa LENDOCARD trabalha com números positivos, lendo os dados como caracteres. Executa-se a conversão para o valor numérico equivalente subtraindo-se o valor do caractere digital do valor do digito 0. Isso vale para qualquer conjunto de caracteres, pois definem-se como contíguos os caracteres 0...9. Note que o programa só possui duas variáveis globais (no "nível 0"). Os dados para o processamento de números e a função VALOR, que realiza a conversão, são usados apenas por LERÇARD e, portanto, declarados e definidos dentro do procedimento. Assim como está, o programa apresenta apenas uma possibilidade de erro — a introdução de um valor cardinal maior do que o MAXINT do compilador invalida a instrução atributiva do loop WHILE.

```
PROGRAM     LENDOCARD     (INPUT, OUTPUT);

TYPE
        CARDINAL    = 0 .. MAXINT;

VAR
        NUMERO      : CARDINAL;
        OK          : BOOLEAN;

        {1111111111111111111111'11111111}

PROCEDURE LERCARD (VAR N    : CARDINAL;
                   VAR OK   : BOOLEAN);
  CONST
        ESPACO      = ' ';
  TYPE
        ORDINAL     = 0 .. 9;
  VAR
        SIMBOLO     : CHAR;
        DIGITOS     : SET OF CHAR;
          {2222222222222222222222222}
                    {NIVEL 2}
  FUNCTION VALOR (DIGITO : CHAR) : ORDINAL;
        {RETORNA VALOR NUMERICO DE DIGITO EM
          QUALQUER CONJUNTO DE CARACTERES}
    BEGIN
      VALOR := ORD (DIGITO) - ORD ('0')

    END; {VALOR}
        {2222222222222222222222222}
              {DE VOLTA AO NIVEL 1}
  BEGIN {LERCARD}

    REPEAT
      READ (SIMBOLO)

    UNTIL SIMBOLO > ESPACO;

    DIGITOS := ['0' .. '9'];

    OK := SIMBOLO IN DIGITOS;

    IF OK THEN
      BEGIN
        N := 0;

        WHILE SIMBOLO IN DIGITOS DO
          BEGIN {CALCULAR NA BASE 10}
            N := 10 * N + VALOR (SIMBOLO);
            READ (SIMBOLO);
          END {QUALQUER DELIMITADOR DE CARACTERES
            OU 'EOLN'}
      END        {APOS O NUMERO E IGNORADO}

  END; {LERCARD}
        {11111111111111111111111111111111}
BEGIN {LENDOCARD - NIVEL 0 - PROGRAMA PRINCIPAL}

  PAGE (OUTPUT);
  WRITELN;
  WRITELN ('ESTE PROGRAMA LE INTEIROS');
  WRITELN ('POSITIVOS (NA FAIXA 0 .. ',
              MAXINT : 1,')');
  WRITELN ('O ENCERRAMENTO DESTE PROGRAMA');
  WRITELN ('E CONTROLADO ATRAVES DA DETECCAO DE
            ERROS');
  WRITELN;
  WRITE ('ENTRE COM UM NUMERO : ');
  LERCARD (NUMERO, OK);

  WHILE OK DO
    BEGIN
        WRITELN;
        WRITE ('O NUMERO DIGITADO FOI : ');
        WRITELN (NUMERO : 1);
        WRITE ('NUMERO ?');
        LERCARD (NUMERO, OK);
    END;

  WRITELN ('... DETECTADO UM ERRO ...' : 40);

END.
```

dores" de tipo para parâmetros sejam identificadores, e não "literais" (0 .. 155, por exemplo).

Estruturas dinâmicas

A criação desse "banco de dados" implica a seguinte modificação: passamos do uso de funções no interior de parâmetros para o desenvolvimento de um algoritmo informal. Além, é claro, do emprego de técnicas precisas de programação para lidar com arquivos de forma organizada — uma exigência fundamental para a elaboração de qualquer banco de dados.

Começaremos pelo exame de algumas variáveis indispensáveis num algoritmo "informal":

```
VAR
  LISTA     : LISTAREG;
  TAMANHO : CARDINAL;
BEGIN
  TAMANHO := 0; {TAMANHO ATIVO DA LISTA}
  {READ DADOS NA LISTA, ATUALIZA
    TAMANHO}
  {CLASSIFIQUE TAMANHO ITENS NA ORDEM
    CORRETA}
  {PRINT TAMANHO ITENS DA LISTA}
END.
```

STUB

Sub-rotina que não efetua qualquer processamento; serve para montar o "esqueleto" do programa, evitando o desenvolvimento explícito de todos os detalhes.

Para traduzirmos esse algoritmo informal em algo mais tangível, exprimimos os três estágios principais do processamento como chamadas de procedimentos (com os nomes apropriados) e relacionamos todos os dados necessários sob a forma de uma lista de parâmetros. Depois de fazer isso, podemos montar um esquema de todos os blocos de procedimentos. Por exemplo, a última instrução do programa será

```
PRINT (LISTA : TAMANHO)
```

Todas as informações essenciais são transferidas ao procedimento na lista de valores de dados e no número de elementos; nesse caso, o cabeçalho não precisará de quaisquer VARs:

```
PROCEDURE PRINT (LISTA : LISTAREG;
                 TAMANHO : LIMITES);
```

Poderíamos prosseguir codificando o procedimento até o final, mas é melhor deixá-lo por enquanto como um "stub" BEGIN-END e lidar com outros procedimentos de "nível 1". Precisamos atribuir um nome para cada um e escolher os itens de dados a serem passados como parâmetros. Além disso, devemos determinar quais procedimentos receberão esses parâmetros e também se algum desses itens será transferido como parâmetro "VAR" (endereço).

Todos os métodos de estruturação de dados examinados até agora possuíam tamanho fixo. Essa especificação vem nas definições TYPE, sendo, portanto, conhecida no momento da compilação. O PASCAL dispõe de estruturas de dados avançadas, que variam de tamanho (e mesmo de tipo, dentro de certos limites) durante a execução do programa. O tamanho de uma estrutura avançada somente é limitado pela capacidade física da memória ou pela disponibilidade do armazenamento de reserva, sendo a estrutura avançada mais comum o arquivo seqüencial.

Assim como as matrizes, cada elemento de um arquivo pode ser de qualquer tipo, simples ou estruturado, com a diferença de que não é possível um arquivo de arquivos. À nossa definição anterior de tipo de registro, poderíamos acrescentar as declarações

```
TYPE
  TIPOARQ = FILE OF DADOS;
VAR
  ARQDADOS : TIPOARQ;
```

que permitem a criação e o processamento de arquivos contendo infinitos componentes, cada qual sob a forma de um registro com nome, débito ou qualquer outro campo desejado.

Da mesma forma que READ (SIMBOLO) serve para a leitura de um único CHAR da entrada de arquivo, usamos READ (ARQDADOS, ITEM) ou WRITE (ARQDADOS, ITEM) para manipular toda uma estrutura de registro como um único item de dados. Caso esses arquivos sejam necessários apenas durante a execução do programa, a única exigência consiste em sua abertura, tanto para leitura como para gravação, por meio dos procedimentos RESET e REWRITE, respectivamente. Para torná-los permanentes, relacionamos os identificadores de arquivo na lista de parâmetros do cabeçalho do programa. Por exemplo,

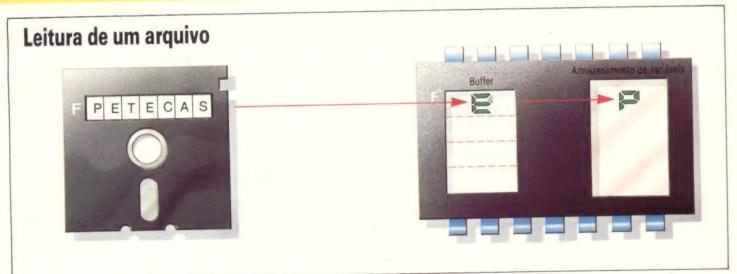
```
PROGRAM DADOS (INPUT, OUTPUT,
  ARQDADOS);
```

Toda vez que usamos as instruções READ ou WRITE, chamamos os procedimentos de entrada e saída predefinidos do PASCAL. Os dois únicos arquivos que conhecemos até agora são, de fato, os dispositivos padronizados de entrada e saída do computador. Em geral, o arquivo INPUT refere-se ao teclado, enquanto OUTPUT sempre é o vídeo dos sistemas de micro. Ao considerar esses dispositivos como arquivos, toda a manipulação de entrada e saída se torna extremamente coerente no PASCAL.

Lembre-se de que, ao digitar WRITE (N), omitindo qualquer nome de arquivo, o valor de N será impresso sob a forma de caracteres no arquivo OUTPUT. A exclusão de um nome de arquivo em instruções READ ou READLN fará com que seja assumido o default — arquivo INPUT, no caso —, isto é, cada "registro" do arquivo será um único valor CHAR.

Arquivos de texto

Os arquivos de texto, ao contrário de outros, possuem, incorporados (além de um marcador de fim de arquivo), marcadores especiais de fim de linha. Estes variam de acordo com o sistema operacional e consistem num único caractere de controle, dois caracteres (CR e LF, por exemplo) ou talvez nenhum, armazenando-se então o tamanho de cada linha.

Leitura de um arquivo

Como garantia de portabilidade, o PASCAL dispõe da função EOLN (F) e dos procedimentos predefinidos READLN (F) e WRITELN (F) — usados apenas com arquivos do tipo texto. Emprega-se a função EOF (F) com qualquer tipo de arquivo. Como o marcador de fim de linha pode ou não ser um único valor CHAR, a leitura desse valor, quando EOLN for verdadeira, fornecerá um caractere Espaço. Assim, devemos sempre verificar EOLN antes. Como os arquivos INPUT e OUTPUT existem independentemente do programa a executar, eles permanecem abertos e não precisam ser localizados ou criados de modo explícito.

No caso de outros arquivos que não INPUT e OUTPUT, temos de atribuí-los a um nome de sistema externo e, então, abri-los para leitura com uma chamada do procedimento RESET — isto é, RESET (UMARQUIVO) — ou criar uma entrada no diretório que prepare a gravação por meio de REWRITE (OUTROARQ). Um esquema geral para o processamento de um arquivo de texto será:

```
{OPEN O ARQUIVO}
WHILE {NAO ATINGIU O EOF} DO
  WHILE {NAO ATINGIU O FIM DA LINHA} DO
    {LE UM CARACTERE}
    {PROCESSA O CARACTERE}
    {SALTA SOBRE O FIM DA LINHA}
```

PUT e GET

Executam-se os procedimentos READ e WRITE com o auxílio de buffers e dos primitivos de entrada e saída PUT (para saída) e GET (para entradas provenientes de arquivos). Quando se abre um arquivo com REWRITE, nenhuma informação entra no buffer até que seja executado um WRITE, o que implica duas operações:

```
F^ := DADOS;
PUT (F)
```

De modo análogo, a instrução READ (F, DADOS) também se exprime por:

```
DADOS := F^;
GET (F)
```

Desse modo, as instruções de entrada e saída em nosso procedimento COPY

```
READ (FONTE, CARACTERE);
WRITE (DESTINO, CARACTERE)
```

também poderiam ser codificadas assim:

```
DESTINO^ := FONTE^;
PUT (DESTINO);
GET (FONTE)
```

Então não precisaríamos requisitar a variável local CHAR, CARACTERE. Talvez seja mais fácil compreender a operação READLN (F) se a expressarmos em termos dos primitivos

```
WHILE NOT EOLN (F) DO
  GET (F); {IGNORA O RESTO DA LINHA, SE
    HOUVER}
GET (F) {E SALTA SOBRE O(S) EOLN(S)}
```

Após um READLN, o buffer do arquivo sempre contém o primeiro item da linha que lerá em seguida. Poderá ser um espaço se EOF (F) for verdadeira. É, portanto, ilegítimo ler um arquivo no caso de EOF ser verdadeira, mas também consistirá num erro realizar qualquer operação, inclusive testes do buffer do arquivo. Por isso, seja cuidadoso ao lidar com outros arquivos que não INPUT e OUTPUT. Essa atenção a condições potenciais de erro garante o desenvolvimento de software extremamente consistente.

Agora que sabemos como "antecipar" um fluxo de dados, passemos a formular o procedimento SALTABRANCOS postulado anteriormente:

```
PROCEDURE SALTABRANCOS (VAR F : TEXT);
  CONST
    ESPACO = ' ';
  VAR
    FEITO : BOOLEAN;
  BEGIN
    FEITO := EOF (F);
    IF NOT FEITO THEN
      FEITO := INPUT^ > ESPACO;
    WHILE NOT FEITO DO
      BEGIN
        GET (F);
```

Na sombra do arquivo
A abertura de um arquivo F no PASCAL implica a criação de uma área buffer correspondente, F^, na qual é lido o primeiro caractere do arquivo. Quando se executa uma operação READ, transfere-se o caractere seguinte do arquivo para o buffer. Se os primeiros caracteres de F forem PETECAS, P será lido em F^. De acordo com a primeira operação READ, P será atribuído a uma variável do PASCAL e substituído por E no buffer F^.

```
        FEITO := EOF (F);
        IF NOT FEITO THEN
          FEITO := F^ > ESPACO
      END
    END; {SALTABRANCOS}
```

Observe que esse programa saltará todos os espaços vazios de um arquivo de texto, inclusive os caracteres de fim de linha. Se queremos apenas saltar o espaço de uma linha qualquer, alteramos a atribuição condicional para

```
FEITO := EOLN (F) OR (F^ > ESPACO)
```

Se, depois, quisermos saltar todos os espaços em branco, faremos

```
REPEAT
  SALTABRANCOS (F);
  IF NOT EOF (F) THEN
    TEMTEXTO := NOT EOLN (F)
UNTIL TEMTEXTO OR EOF (F)
```

Com essa modificação fica fácil desenvolver programas interativos que verificam entradas nulas. Por exemplo,

```
REPEAT
  WRITE ('ENTRE COM DADOS : ');
  SALTABRANCOS (INPUT)
UNTIL NOT EOLN (INPUT)
```

Esse programa será invalidado se introduzirmos o caractere de controle usado pelo sistema para indicar o final do arquivo; mas sempre é possível empregar o esquema anterior para tornar o programa confiável.

Considera-se o procedimento ASSIGN uma "extensão padronizada", e é provável que em breve esteja oficialmente adotado, junto com OPEN e SEEK, para arquivos de acesso aleatório. Outras extensões importantes são

```
FUNCTION FSTAT (FILENAME)
```

que fornece um resultado booleano — verdadeiro, se um arquivo existe — e o recurso

```
PROCEDURE RENAME (OLDNAME, NEWNAME)
```

Variações do PASCAL

No PASCAL, não existem restrições quanto ao número de arquivos que você pode abrir ao mesmo tempo. No entanto, como eles exigem o uso do sistema operacional do computador, essa é uma área onde ocorrem divergências em relação às estritas regras padronizadas. Por exemplo, algumas versões da linguagem não admitem arquivos.

As convenções do sistema operacional relativas à atribuição de nomes e arquivos ocuparão cerca de dez caracteres de qualquer identificador de arquivo. Desse modo, nosso exemplo criaria um arquivo permanente denominado ARQDADOS. Em muitos sistemas, as convenções para atribuição de nomes são tais que isso se torna impossível.

A "extensão padronizada" para vinculação de um identificador de arquivo no PASCAL é o procedimento de atribuição, usado antes de RESET ou REWRITE:

```
ASSIGN (IDENTARQ, NOMESTRING);
```

Assim, para nossos objetivos, ASSIGN (ARQDADOS, 'D : NOSSOARQ.DAD') seria mais do que suficiente.

Programa Copiatexto

O PASCAL dispõe do identificador TEXT para o tipo mais comum de arquivo. Esse identificador é usado na declaração de qualquer variável do arquivo de texto, na lista de parâmetros do cabeçalho, no início do programa. Para tornar o programa mais útil em termos gerais, mostramos o recurso de cópia do arquivo de texto, formulado no procedimento COPY (capaz de processar dois arquivos quaisquer). Os compiladores não padronizados exigirão o uso de RESET (F1, 'FONTE') e assim por diante, no programa principal. Transferem-se a ambos os arquivos como parâmetros VAR para o procedimento COPY. Deve-se isso tanto à desatualização do arquivo a que se destina como ao fato de que o arquivo FONTE será lido e, assim, seu estado se modificará. Por isso, as variáveis de arquivo sempre são transferidas por endereço, nunca por valor. Além de outras considerações, um parâmetro de valor implica uma cópia local — que, junto com os arquivos, sempre ocupa potencialmente mais espaço do que o disponível na memória.

```
PROGRAM     COPIATEXTO     (F1, F2);

VAR
    F1,
    F2      : TEXT;

{1111111111111111111111111111111}

PROCEDURE COPY (VAR FONTE,
                    DESTINO : TEXT);
  VAR
      CARACTERE     : CHAR;
  BEGIN
    WHILE NOT EOF (FONTE) DO
      BEGIN {COPIA UMA LINHA : }
        WHILE NOT EOLN (FONTE) DO
          BEGIN
            READ (FONTE, CARACTERE);
            WRITE (DESTINO, CARACTERE)
          END;
            {COPIA O EOLN}
        READLN (FONTE);
        WRITELN (DESTINO);
      END

  END;    {COPY}
{1111111111111111111111111111111}

BEGIN     {COPIATEXTO - PRINCIPAL}

  ASSIGN (F1, 'FONTE');
  RESET (F1);       {LOCALIZA E ABRE PARA LEITURA}
  ASSIGN (F2, 'DESTINO');
  REWRITE (F2);     {CRIA O ARQUIVO}
  COPY (F1, F2)

END.
```

N1684

Projetado e desenvolvido pela Scopus, o Nexus 1684 presta-se tanto para uso pessoal como profissional. Compatível com o IBM PC, inclui comandos para geração de gráficos, de cor e de som.

A família Nexus 1600 compreende o N1614, o N1615 e o N1684 — microcomputadores projetados e construídos no Brasil, totalmente compatíveis com o IBM PC.

Recebendo o microprocessador 8088 de 16 bits, com opção para co-processador aritmético 8087, a unidade de sistema abriga três placas e um acionador de disquete (no N1614) ou duas (no N1615), elevando a capacidade de 320 para 640 Kbytes. A versão mais recente, o N1684, conta com uma unidade de disquete e uma de disco rígido tipo Winchester de 10 Mbytes, slim (ocupando metade da altura das outras). Essas são as diferenças básicas entre os três modelos.

O teclado do Nexus 1600 é serial, tipo QWERTY, com 86 teclas (dez de função), bloco numérico separado, simulador de ruído mecânico e repetição automática. O design do teclado engloba detalhes que facilitam sua operação: a conexão à unidade de sistema através de cabo espiralado, que aumenta sua mobilidade; e as dimensões reduzidas, leveza e pés retráteis, que conferem maior flexibilidade no uso diário.

O monitor de vídeo possui cinescópio móvel, adaptável ao ângulo de visão do operador. O Nexus 1600 oferece duas opções de vídeo. Uma delas, o monitor monocromático, de 12 polegadas, em fósforo verde, representa as cores virtuais em tonalidades distintas. É controlado por uma interface gráfica que opera em modo alfanumérico (80 x 25 ou 40 x 25) e gráfico branco e preto (640 x 220 pixels), além de aceitar conexão para caneta óptica. A outra opção consiste num monitor de vídeo colorido, de 14 polegadas, controlado pela mesma placa gráfica já descrita, oferecendo, nesse caso, um modo alfanumérico colorido (dezesseis cores, 80 x 25 ou 40 x 25), ou gráfico colorido (quatro cores, 320 x 200 pixels) e, ainda, o mesmo modo gráfico monocromático (640 x 200 pixels). O monitor dispõe também de máscara anti-reflexo, que elimina cintilações indesejáveis.

Uma placa de funções múltiplas engloba interfaces para joystick, para discos flexíveis de 5 1/4 polegadas e relógio-calendário não volátil, nos dois primeiros modelos. O Nexus 1600 dispõe de mais quatro placas de expansão: a primeira é uma interface para miniwinchester, de 10 Mbytes (específica para o N1614 e o N1615); a segunda, a Incox, para cabo co-axial IBM 327x, possibilita ligação direta com controladoras IBM 3274 e 3276, emulando os terminais IBM 3278 e 3279; as outras são saídas para impressora paralela e expansão de RAM.

Solução integrada
Lançado em 1985, o modelo 1684 da família Nexus já vem equipado com uma unidade de disco do tipo Winchester, com capacidade de 10 Mbytes.

O sistema operacional do Nexus 1600 — o SISNE — foi escrito em PASCAL e, na versão inicial, emula o PC-DOS 1.1 da IBM. Quanto aos aplicativos, o usuário pode utilizar os inúmeros pacotes já existentes no mercado brasileiro. Entre eles, incluem-se os destinados à comunicação de dados: o NX-830, para integração com os sistemas de grande porte da Burroughs; o NX-VIP, para os computadores Honeywell Bull; o NX-Z, de uso geral; e o NX-3270, para IBM.

Assim, interligado em rede com outros computadores, o Nexus 1600 compartilha arquivos e periféricos, reduzindo custos e agilizando operações de processamento. Outra vantagem desse micro está na possibilidade de multiprogramação, que permite executar vários programas simultaneamente.

Solução racional
Além da placa de sistema — que compreende um processador Intel 8088 e duas interfaces de comunicação serial — o Nexus 1684 conta ainda com placas específicas para o controle das unidades de disquete, disco Winchester e monitor de vídeo.

Placa controladora das unidades de disco

Fonte de alimentação

Unidade de disco tipo Winchester

NEXUS 1684

MICROPROCESSADOR

Processador central 8088; co-processador aritmético de ponto flutuante (opcional) 8087.

CLOCK

8 MHz.

MEMÓRIA

256 Kbytes de RAM, expansível até 704 Kbytes; 64 Kbytes de ROM.

VÍDEO

Monitor de 12": monocromático, em fósforo verde, com capacidade multinível (múltiplas tonalidades). Modo alfanumérico, 80 x 25 ou 40 x 25 caracteres; modo gráfico, 640 x 200 pixels. Monitor de 14": em cores. Modo alfanumérico: 640 x 200 pixels, dezesseis cores; modo gráfico: 320 x 200 pixels, quatro cores.

TECLADO

Tipo máquina de escrever com 86 teclas (dez de função), conjunto numérico separado, repetição automática e simulador de ruído mecânico.

LINGUAGEM

BASIC incorporado.

PERIFÉRICOS

Unidades de disco (5 1/4" e miniwinchester), impressora, interface serial RS232C, joystick e caneta óptica.

DOCUMENTAÇÃO

Quatro manuais acompanham o equipamento (apresentação, referência técnica, operação e referência BASIC, fornecendo uma visão das aplicações possíveis do sistema, de sua constituição e das possibilidades de expansão. Além das explicações sobre o modo de operação, informações adicionais orientam a respeito da programação em BASIC.

OS NOMES DOS NOMES

Os gerenciadores de bancos de dados manipulam informações, mas é necessário organizá-las para que o sistema seja útil e eficiente. A importância do planejamento aparece logo nos primeiros estágios.

Para quem usa computadores, a afirmação pode parecer imprópria, mas a praticidade dos livrinhos de endereços como bancos de dados pessoais é impressionante. São até portáteis! Sua única exigência fundamental: que os registros estejam nas páginas destinadas à letra inicial do nome de cada pessoa.

A flexibilidade é tanta que pode haver registros do tipo:

João Enéias, r. França, 316, Vila Municipal, CEP 13300, Piracicaba, SP, tel.: 83-1511 — tel. consultório do pai (dr. Alfredo): 88-2115, só pela manhã; — à tarde, ligar para 88-2128; muda em jan.

E, mais à frente, uma indicação como esta:

Vilaça (Maurinho), r. Moura.

Ou, ainda, assim:

Psiquiatra: 210-6095.

Quem usa um livro de endereços tem liberdade para ordenar os dados da maneira que achar melhor: por nome, sobrenome ou qualquer outro indicador que achar adequado. Isso funciona muito bem no papel, mas, quando se passa um banco de dados como esse para um computador, a abordagem precisa ser melhor sistematizada.

Tomando-se por princípio que um livro de endereços eletrônico terá algumas desvantagens em relação ao original, podemos criá-lo de tal modo que seja pelo menos suficientemente flexível para nos atender em várias situações. Para isso, vamos examinar campo por campo os registros que iremos usar, começando pelo do nome.

Nomes de pessoas têm duas partes: a chamada "metade genérica" (o sobrenome) e a "metade específica" (o nome de batismo). Para certas circunstâncias, convém tomar conhecimento de que, em países como Japão, China e Hungria, o sobrenome vem escrito antes do nome.

Será preciso prever, ainda, a extensão máxima que vai ocupar cada registro. Se o sistema de banco de dados tiver comprimento de campo fixo, imagine todas as situações possíveis — por exemplo, a entrada de sobrenomes de dezoito letras.

Como teremos, de qualquer maneira, de impor um formato, é preferível usar o campo do sobrenome como chave primária de localização do registro.

Depois, vem o problema do endereço. Você já tentou preencher um formulário onde não há espaço para o número do seu apartamento? Pois o problema aqui se assemelha. Se você tiver de lidar com endereços do exterior, a situação pode ficar ainda mais complicada. No Japão, só para citar um exemplo, quando se escreve um endereço, o nome da cidade vem em primeiro lugar. O nome da pessoa é indicado por último.

Especificações básicas

Tanto neste banco de dados que montaremos como exemplo quanto em outros, mais complexos, será praticamente impossível atender a todos os imprevistos. Mas convém admitir pelo menos algumas premissas. Por exemplo: é pouco provável que você tenha de manter nesse arquivo um endereço da Coréia ou do Japão, ou que precise registrar alguém com dezesseis nomes. Então, a estrutura básica poderia ser:

- Campo do sobrenome — até 40 caracteres;
- Campo do nome — até 60 caracteres;
- Campo de endereço, 1.ª linha — até 80 caracteres;
- Campo de endereço, 2.ª linha — até 80 caracteres;
- Campo de endereço, 3.ª linha — até 80 caracteres;
- Campo de endereço, 4.ª linha — até 80 caracteres;
- Campo de endereço, 5.ª linha — até 80 caracteres;
- Campo do telefone — até 20 caracteres; e
- Campo para observações — até 80 caracteres.

Uma especificação básica de registro como essa deve cobrir a maior parte dos imprevistos. Assim, pode haver dificuldades com o tamanho limitado do campo das observações. Outro problema: se você precisa fazer pesquisas por país, então esse campo deve estar separado. Decida-se a respeito ainda na fase inicial do sistema.

O software gerenciador de banco de dados é que determinará a facilidade com que se farão essas mudanças. Este artigo toma como exemplo um gerenciador muito prático, o Card Box (caixa de fichas). O programa tem limitações em

seus recursos de acesso e manipulação de dados, mas, para quem está mais interessado em rapidez sem muita confusão, os resultados satisfazem.

Ele não oferece, como outros gerenciadores, uma linguagem de programação para manipular arquivos ou registros. Em compensação, permite que você recupere registros usando comandos bem simples. Para criar o banco de dados que explicamos a seguir, os comandos também são muito simples.

Usando o Card Box

A primeira coisa a fazer, neste ou em qualquer outro banco de dados, é escolher um formato para os registros. Isso vai determinar que informações deverão ser armazenadas, sua indexação (isto é, que campos consideraremos chaves para a busca dos registros) e como esses registros aparecerão na tela ou nas listagens da impressora.

Se batizarmos nosso banco de dados com o nome de ENDERECOS, o gerenciador imediatamente poderá criar um arquivo de formatos chamado ENDERECOS.FMT. Esse arquivo é editável, ou seja, podemos modificar a maneira como as informações se apresentam — e até criar outros arquivos de formatos.

Como a maioria dos programas desse tipo, o Card Box permite a inserção de textos permanentes para a descrição de cada campo. Imagine ver no vídeo indicações como estas:

```
06168
3995
86
185.000
bico fino
```

Elas ficariam muito mais claras se os campos estivessem identificados deste modo:

```
NUMERO DO FABRICANTE        06168
NOSSO NUMERO                 3995
ESTOQUE ATUAL                  86
PRECO                     185.000
MODELO                  bico fino
```

Nos dois casos, as informações são absolutamente as mesmas, mas no segundo minimizou-se a possibilidade de confusão. Para efeito de pesquisa, o programa permite que o usuário dê um destes atributos a cada campo: NADA, MAN(ual), AUTO e TODOS. Num banco de dados de estoque, por exemplo, dificilmente alguém precisaria pesquisar por PRECO. Será bem mais provável que um almoxarife precise saber quantos artigos de número 06168 ainda estão disponíveis para venda. Assim, o campo NUMERO DO FABRICANTE pode ser chave.

Se estivermos usando um banco de dados que contenha os registros sobre empresas e pessoas, precisaremos de buscas por nome, sobrenome, cidade e, às vezes, até pelo código telefônico.

O mais importante de tudo é entender que não se pode criar um bom sistema de banco de dados enquanto não se souber exatamente de que maneira ele será usado. Imagine que você mandasse imprimir fichas de cartolina para arquivá-las numa caixa e ter nelas seus registros. É mais ou menos a mesma coisa quando você projeta seu banco de dados: se não fizer uma previsão correta de suas necessidades, isso poderá lhe custar tempo, dinheiro e aborrecimento.

Identificadores
Estes identificadores determinam a ordem de acesso aos registros. Eles tornam possível, por exemplo, a classificação de todos em ordem alfabética, por meio de um comando do programa. Essa classificação pode ser feita por nome, sobrenome, cidade, CEP ou qualquer outro campo escolhido.

Comprimento do campo
Campos grandes facilitam o uso de seu sistema de banco de dados; por outro lado, quanto maiores forem, menos registros poderão ser colocados em cada arquivo. Em geral os campos têm seu comprimento demarcado por traços ou outros caracteres gráficos.

Formato e conteúdo
Acima vemos a definição de um formato para os registros de um banco de dados. A linha inferior mostra os comandos disponíveis nessa fase. A tela abaixo mostra o conteúdo de um registro, com opções para pesquisas e alterações necessárias.

Identificação do campo
Você pode usar os nomes que escolher para identificar cada campo, e faz isso ao projetar o formato do registro.

Companhia Agricola Piracicaba
Endereco: r. Franca, 316
Cidade Piracicaba CEP 13.300
Telefone (011) 83-1511
Contatos Joao Eneias Jose Carlos
Palavras-chaves Tratores/Pas
Produtos recebidos Pas
Pesquisa: I(ndice), A(pagar), L(istar), C(have), M(odificar), F(im)

NA VELOCIDADE DA LUZ

Funcionamento
Um gravador de EPROM insere nos chips programas de uso freqüente. Eles são primeiro carregados na memória do micro e depois transferidos ao gravador.

A demora na execução de tarefas como carregar programas do disco ou da fita pode ser evitada com o gravador de EPROMs — dispositivo de grande utilidade que tende a se popularizar entre os usuários.

Uma das coisas que mais fascinam no primeiro contato com os computadores é sua velocidade: ao simples toque de um tecla, várias coisas acontecem imediatamente. Mas, antes disso, a pessoa que opera o computador precisa executar certas tarefas, as quais, diante da rapidez de alguns programas, parecem lentas demais.

Quando o micro opera com unidades de disco, é necessário ligá-lo, abrir o drive, colocar o disquete, carregar o sistema operacional e, só depois disso, inserir o programa desejado. No caso de equipamentos que operam com fita cassete, o sistema operacional e a linguagem BASIC gravados em ROM estão disponíveis quase de imediato, mas o carregamento de qualquer programa parece uma operação infindável, além de sujeita a erros.

A melhor solução para agilizar esse processo é gravar os programas em EPROM (Erasable Programmable Read Only Memory, "memória programável apenas para leitura e apagável") e utilizá-la do mesmo modo que os cartuchos de videojogos (ou inseri-la nos conectores sobressalentes encontrados em vários micros). Assim, um simples comando faz com que o processador de texto ou outro programa gravado em EPROM seja carregado na memória do micro, a uma velocidade tão alta que parecerá instantânea.

A rigor, é possível gravar em EPROM qualquer espécie de programa. Vários fabricantes mantêm verdadeiras coleções de linguagens e outros utilitários em EPROM para vender a seus clientes: processadores de texto, planilhas de cálculo, linguagem LOGO etc. E você mesmo armazena nelas os programas que usa com maior freqüência.

Como toda ROM, a EPROM é uma densa malha de condutores elétricos, dispostos em linhas e colunas, como uma matriz, onde cada interseção constitui uma "célula". Se numa delas há conexão entre dois condutores, o estado lógico naquele ponto será 1; do contrário, será 0. Quando uma série de impulsos elétricos correspondentes a um endereço chega ao bus de endereçamento do micro, a eletricidade passa em oito desses condutores (os oito impulsos representam 1 byte). Dependendo da existência de conexão nos cruzamentos, a corrente elétrica será reconduzida ao bus de dados do micro. Dessa maneira, transferem-se os dados de determinado endereço para a CPU.

Em geral, os estados lógicos numa EPROM partem todos do valor 1 — cada endereço mantém o valor hexadecimal FF. Mas, se uma corrente de voltagem elevada passa por um célula, a conexão se rompe, criando, dessa forma, o estado lógico 0.

Para gravar 1 byte num endereço qualquer da EPROM, envia-se antes, aos pinos corretos, o conjunto de bits especificador do endereço que se quer gravar. Esse conjunto é transmitido pelo bus de endereçamento. Em seguida, um pulso de 25 V atua por 50 microssegundos sobre os pinos em que se quer estado 0. E o endereço fica gravado. O processo repete-se com os endereços seguintes, até que a EPROM inteira esteja gravada. Em outros tipos programáveis de ROM esse processo não permite reversão, mas uma EPROM é por definição apagável.

A EPROM caracteriza-se por uma pequena janela de quartzo sobre o chip, através da qual se pode vê-lo. Quando uma luz ultravioleta forte atravessa a janela, os átomos dos condutores metálicos que formam a malha se ionizam, e a ligação rompida pela passagem da corrente de 25 V se restabelece.

Isso permite que a corrente volte a circular em todos os cruzamentos da malha de condutores.

Desse modo, o estado lógico torna-se novamente 1 em todos os endereços. A luz ultravioleta não é essencial para apagar a EPROM, já que se pode usar um gravador de EPROM com uma rotina de software para registrar o byte FF (composto apenas por dígitos 1) em todas as posições de memória.

Um bom aparelho desse tipo permite a gravação de até 16 Kbytes de dados e instruções numa EPROM. Os gravadores de EPROM vêm, quase sempre, acondicionados numa pequena caixa metálica, cuja característica que mais se sobressai é um conector ZIF (Zero Insertion Force, "força de inserção zero"). Nele se inserem os chips sem risco de danos para os pinos — que, se quebrados, inutilizam o componente. O ZIF recebe EPROMs de 24 ou 28 pinos. Em alguns modelos, existem dois conjuntos de oito chaves alteradas conforme as características da gravação.

A chave que seleciona as operações de leitura ou gravação da EPROM constitui outro importante dispositivo desses aparelhos. Quando se usa o gravador, é preciso ter certeza de que a chave encontra-se na posição correta; caso contrário, uma EPROM que devia apenas ser lida acaba regravada, o que provocaria o apagamento do programa nela contido.

Em geral, a ligação entre micro e gravador se faz por um cabo flexível multivias, saída serial RS232C ou placa periférica. O programa que vai ser passado para a EPROM fica numa área de buffer da memória do micro, e um software específico auxilia a transferência. Se ele está em cassete, a primeira coisa a fazer é copiá-lo numa EPROM.

Depois de carregados, quase todos os softwares apresentam na tela um menu com as opções disponíveis. Por exemplo: para copiar em EPROM, carregue o programa a ser gravado utilizando a opção L (de load). Esse comando o coloca na área de buffer criada a partir de um endereço determinável. Depois, selecione a opção P (de programas) do menu principal. A tela exibe então a lista dos tipos de EPROM programáveis para escolha. Feita a opção, surgem no vídeo as posições em que devem ser colocadas as dezesseis chaves, quando necessário.

Assim que todas estiverem na posição correta, o programa dará instruções para que a chave seletora de leitura/gravação seja colocada na posição "gravar". Em geral, depois, basta apertar qualquer tecla, que a transferência tem início. À medida que cada posição de memória vai sendo gravada, o número de endereço correspondente aparece no vídeo. O tempo gasto no processo depende da quantidade de endereços a gravar: desde uns poucos segundos (em caso de programa curto) até 14 minutos (para uma EPROM de 16 Kbytes).

Os bons programas de gravação têm vários recursos. Um dos mais úteis é o comando que verifica se a gravação de cada endereço está correta e lista na tela os de conteúdo errado. Não havendo erros, a EPROM pode ser utilizada com segurança.

Conexão ao micro
Em geral, por cabo flexível multivias, saída serial RS232C ou, no caso do Apple, por placa periférica

Em vista da grande utilidade desses dispositivos, surpreende verificar que os gravadores de EPROM ainda não sejam populares entre os proprietários de micros. Talvez porque muitos usuários considerem o uso desse aparelho restrito ao "sério" mundo dos profissionais da informática. Puro engano, pois os gravadores de EPROM são fáceis de operar e não exigem qualquer conhecimento de eletrônica. Oferecem, ainda, uma série de vantagens, como a grande velocidade de acesso aos aplicativos e a possibilidade de estes serem chamados pelos programas normais em BASIC.

Pino a pino

Diagrama do chip EPROM TMS27128, de 16 Kbytes, com quatorze pinos para endereçamento e oito para saída dos dados armazenados.

Nomes dos pinos	
A [illegible] A	[illegible]
CE	[illegible]
OE	[illegible]
O [illegible] O	[illegible]
PGM	[illegible]
Vcc [illegible]	[illegible]
Gnd	[illegible]

Sinal	Pino	Pino	Sinal
Vpp	1	28	Vcc
A	2	27	PGM
A	3	26	A
A	4	25	A
A	5	24	A
A	6	23	A
A_3	7	22	OE
A_2	8	21	A
A_1	9	20	CE
A_0	10	19	O
O_0	11	18	O
O	12	17	O
O	13	16	O
Gnd	14	15	O

A EPROM em close
Pela janela de quartzo podem-se ver os delgados filamentos que ligam o chip aos pinos de seu invólucro e a malha de condutores metálicos (matriz de memória) onde a informação fica registrada.

BUSCA DE VARIÁVEIS

Você mesmo pode criar programas que dispõem de recursos auxiliares de edição ou outros subsídios à tarefa de programação em microcomputadores — são os chamados "utilitários".

Os programas utilitários que vêm incorporados ao BASIC dos microcomputadores variam consideravelmente de fabricante para fabricante. Alguns micros, como TK 82C, TK 83, TK 85, CP 200, Ringo, TK 90X, Unitron, Maxxi, TK 2000 etc., possuem um simples editor de linhas. Outros, como os da linha TRS-80 (CP 500, Sysdata, CP 400 Color), possuem recursos adicionais, extremamente úteis, como AUTO (para numeração automática das linhas) ou RENUM (para renumerar as linhas de programas já existentes).

Os utilitários são geralmente escritos em código de máquina, pela rapidez que ele proporciona e também por não ser nada fácil um programa em BASIC alterar a si mesmo. No entanto, alguns utilitários bem simples podem ser escritos em BASIC. Desse modo, vamos concentrar a atenção no funcionamento e na operação dos utilitários, sem considerar detalhes mais complicados, como as ações do sistema operacional e o próprio código de máquina.

Nos micros Sinclair e Apple, cada linha de programa começa com 4 bytes: dois dedicados ao número e dois ao comprimento (nos TKs) ou ao endereço do início da próxima linha (nos Apples). Após esses bytes iniciais, vem o texto, em BASIC, codificado com as palavras reservadas, os caracteres especiais, os nomes de variáveis, funções etc. Por fim, existe 1 byte destinado a assinalar o final da linha.

Estrutura de uma linha em BASIC

Primeira linha | Outras linhas

1.º byte | 2.º byte | 3.º byte | 4.º byte | bytes com o texto em BASIC | último byte

No TK 85: número da linha = 256 x (1.º byte) + (2.º byte)
comprimento da linha = (3.º byte) + 256 x (4.º byte)
último byte = 118.

No TK 90X: número da linha = 256 x (1.º byte) + (2.º byte)
comprimento da linha = (3.º byte) + 256 x (4.º byte)
último byte = 13.

No Apple: número da linha = (3.º byte) + 256 x (4.º byte)
endereço do 1.º byte da próxima linha = (1.º byte) + 256 x (2.º byte)
último byte = 0.

Embora seja difícil um programa em BASIC se modificar, não há grandes problemas em fazer com que inspecione outro programa em BASIC. O utilitário apresentado a seguir, nas versões para as linhas Sinclair e Apple, com variações para o TK 90X, pesquisa sistematicamente um programa em BASIC procurando o nome de uma variável (ou função, no caso do TK 90X e do Apple). Os números das linhas em que ele se encontra são mostrados no vídeo.

O utilitário começa determinando onde tem início, na memória do micro, o texto do programa em BASIC a ser pesquisado. A seguir, percorre o programa linha por linha, saltando os trechos sem nomes de variáveis e extraindo os que encontra. Finalmente, compara os nomes extraídos do programa com aquele que está sendo procurado.

Quando inicia a pesquisa de uma nova linha do texto em BASIC, o utilitário registra o número desta (que é armazenado em 2 bytes) e seu comprimento (nos Sinclairs) ou o número do byte do início da próxima linha (nos Apples). Para maior clareza, observe nos programas ilustrados as estruturas de armazenamento para essas linhas de microcomputador.

O programa salta as linhas REM e as seqüências de caracteres entre aspas, onde em geral não se encontram nomes de variáveis. Os números armazenados em 5 bytes após comandos GOTO ou GOSUB ou atribuições de variáveis também são ignorados.

Depois disso, o programa segue inspecionando cada linha em busca de nomes de variáveis. E estes têm, necessariamente, como caractere inicial, uma letra. Os demais caracteres, quando existem, são letras ou números. As variáveis alfanuméricas apresentam como último caractere o cifrão ($). As matrizes e as funções (no TK 90X e no Apple) têm seus nomes seguidos por parênteses. Considerando essas condições, o programa distingue funções e variáveis numéricas, alfanuméricas e indexadas.

No caso do TK 90X, há outras complicações. As variáveis podem ter nomes em letras maiúsculas, minúsculas ou ambas. Desse modo, as variáveis de nome ALEPH, aleph e Aleph são tratadas como uma só. Portanto, a versão para o TK 90X converte todas as minúsculas em maiúsculas antes de fazer a comparação. O programa para TKs não faz distinção entre matrizes e variáveis alfanuméricas simples. Desse modo, uma matriz T$(4) e o quarto caractere de um string (T$(4)) são considerados a mesma coisa. Mas isso não constitui uma limitação real, pois o BASIC desse equipamento não permite uma matriz e uma variável alfanumérica simples identificadas pelo mesmo nome.

O utilitário deve ser inicialmente digitado e gravado; depois, carregado junto com o programa que se quer pesquisar. O TK 85 e o Apple

necessitam de outros programas auxiliares (os MERGEs), existentes em várias versões, que juntam dois ou mais programas num só.

O TK 90X já incorpora um comando MERGE, bastando carregar o utilitário e aplicar esse comando e o nome do programa a ser pesquisado.

Nos TKs, deve-se comandar RUN 9700 para a execução e, nos micros da linha Apple, GOTO 30000. O utilitário pedirá o nome da variável que deve procurar. Se este for o de uma matriz ou função (para o TK 90X), acrescente um parêntese aberto — (— ao final do nome.

Apple e TK 2000

```
30000 INPUT "QUAL VARIAVEL
      DEVO PROCURAR ?";OBJ$
30010 O1=LEN (OBJ$)
30020 EA=2049
30025 REM ACHA END. PROX.
      LINHA
30030 E1= PEEK ( EA )+256 *
      PEEK ( EA+1 )
30035 REM ACHA NUMERO DA LINHA
30040 NL= PEEK ( EA+2 )+256 *
      PEEK ( EA+3 )
30050 IF NL = 30000 THEN END
30060 CEA= EA + 4
30065 REM COMECA A PROCURAR A
      VARIAVEL NA LINHA
      DO PROG.
30070 FOR X = CEA TO E1
30080 P=PEEK (X)
30085 REM SE FOR REM OU DATA
      PULA PARA A PROX. LINHA
30090 IF P=178 OR P=131 THEN
      EA=E1: GOTO 30030
30095 REM SE FOR ASPAS, VAI
      PARA SUB-ROTINA
30100 IF P=34 THEN GOSUB 30230
30105 REM COMECA A COMPARAR
      LETRA POR LETRA P/
      ACHAR VARIAVEL
30110 CO=0
30120 R=1
30130 FOR N = X TO X + O1 -1
30140 IF ASC( MID$(OBJ$,R,1))=
      PEEK (N) THEN CO=CO+1
30150 R=R+1
30160 NEXT N
30165 REM SE ACHAR VARIAVEL
      VAI PARA 30190
30170 IF CO=1 AND PEEK
      (N-1-O1)>64 AND PEEK
      (N-1-O1)<91 AND(N-1-O1)
      <>(EA+3) THEN GOTO 30190
30180 IF CO=O1 THEN GOTO 30280
30185 REM SE NAO,VOLTA E COME-
      CA UMA LETRA A FRENTE
30190 CEA=CEA+1
30195 REM VE SE NAO E' FIM DA
      LINHA
30200 IF CEA=E1-1 THEN EA=E1 :
      GOTO 30030
30210 NEXT X
30220 EA=E1 : GOTO 30030
30225 REM PROCURA PROXIMA
      ASPAS. ENCONTRANDO,VOLTA
30230 X=X+1
30240 P=PEEK (X)
30250 IF P=51 AND X+2=E1 THEN
      POP : GOTO 30030
30260 IF P=34 THEN RETURN
30270 GOTO 30230
30275 REM IMPRIME O NUMERO DA
      LINHA SE POSSIVEL
30280 P=PEEK (N)
30290 IF P>199 AND P<210 THEN
      PRINT NL
30300 IF P=44 OR P=58 OR P=0
      OR P=198 OR P=59 THEN
      PRINT NL
30310 X=N
30320 P=PEEK (X)
30330 IF X= E1 THEN EA=E1 :
      GOTO 30030
30335 REM PROCURA DOIS PONTOS
30340 IF P=58 THEN GOTO 30190
30350 X=X+1 : GOTO 30320
```

TK 85 e CP 200

```
9700 PRINT "ENTRE A VARIAVEL
9705 INPUT T$
9710 [illegible] T$
9715 LET [illegible]
9720 LET [illegible]
9725 LET NL=118
9730 LET NUM=126
9735 LET PROG=16509
9740 LET TP=PROG
9745 LET LIN=256*PEEK (TP)+PEEK
(TP+1)
9750 IF LIN>=9000 THEN STOP
9755 LET TP=TP+2
9760 LET CL=PEEK (TP)+256*PEEK
(TP+1)
9765 LET TP=TP+2
9770 LET PL=TP+CL
9775 IF PEEK (TP)<>NL THEN GOTO
9790
9780 LET TP=TP+1
9785 GOTO 9745
9790 IF PEEK (TP)<>REM THEN GOTO
 9810
9795 REM ■PULA LINHAS REM
9800 LET TP=PL
9805 GOTO 9745
9810 IF PEEK (TP)<>ASPAS THEN GO
TO 9855
9815 REM ■PULA TUDO ENTRE ASPAS
9820 LET TP=TP+1
9825 IF PEEK (TP)<>NL THEN GOTO
9840
9830 LET TP=TP+1
9835 GOTO 9745
9840 IF PEEK (TP)<>ASPAS THEN GO
TO 9820
9845 LET TP=TP+1
9850 GOTO 9775
9855 IF PEEK (TP)<>NUM THEN GOTO
 9880
9860 REM ■PULA CINCO BYTES COM
          NUMEROS BINARIOS
9865 LET TP=TP+6
9870 GOTO 9775
9875 REM ■1º CARACTERE DO NOME
         DEVE SER UMA LETRA
9880 IF PEEK (TP)<CODE "A" OR PE
EK (TP)>CODE "Z" THEN GOTO 9895
9885 LET C$=CHR$ (PEEK (TP))
9890 GOTO 9905
9895 LET TP=TP+1
9900 GOTO 9775
9905 LET N$=""
9910 REM ■GUARDA NOME DE VARIAVE
L
9915 LET N$=N$+C$
9920 LET TP=TP+1
9925 REM ■VERIFICA SE O NOME ACA
B[illegible]
9930 IF PEEK (TP)<CODE "0" OR PE
EK (TP)>CODE "Z" THEN GOTO 9945
9935 LET C$=CHR$ (PEEK (TP))
9940 GOTO 9915
9945 REM ■TERMINA COM $ PARA
         VARIAVEIS STRINGS
9950 IF PEEK (TP)<>CODE "$" THEN
 GOTO 9970
9955 LET N$=N$+"$"
9960 LET TP=TP+1
9965 GOTO 9990
9970 REM ■TERMINA COM ( PARA
            MATRIZES
9975 IF PEEK (TP)<>CODE "(" THEN
 GOTO 9990
9980 LET N$=N$+"("
9985 LET TP=TP+1
9990 IF N$=T$ THEN PRINT N$;" NA
 LINHA ";LIN
9995 GOTO 9775
```

TK 90X

Alterações a serem feitas no programa do TK 85 para que ele rode no TK 90X

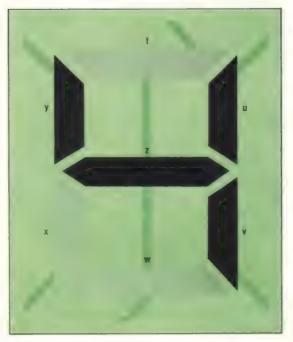
```
9706 FOR i=1 TO LEN (T$)
9707 IF T$(i)>="a" AND T$(i)<="z"
 THEN LET T$(i)=CHR$ (CODE T$(
i)-32)
9708 NEXT i

9720 LET ASPAS=34
9725 LET NL=13
9730 LET NUM=14
9735 LET PROG=23635
9740 LET TP=PEEK (PROG)+256*PEEK
 (PROG+1)

9880 IF PEEK (TP)>=CODE "A" AND
PEEK (TP)<=CODE "Z" THEN LET C$=
CHR$ (PEEK (TP)): GOTO 9905
9885 REM ■SUBSTITUI MINUSCULAS
            POR MAIUSCULAS
9890 IF PEEK (TP)>=CODE "a" AND
PEEK (TP)<=CODE "z" THEN LET C$=
CHR$ (PEEK (TP)-32): GOTO 9905

9930 IF PEEK (TP)>=CODE "A" AND
PEEK (TP)<=CODE "Z" THEN LET C$=
CHR$ (PEEK (TP)): GOTO 9915
9935 REM ■SUBSTITUI MINUSCULAS
            POR MAIUSCULAS
9940 IF PEEK (TP)>=CODE "a" AND
PEEK (TP)<=CODE "z" THEN LET C$=
CHR$ (PEEK (TP)-32): GOTO 9915
9942 REM ■VERIFICA SE O NOME TEM
              NUMEROS
9943 IF PEEK (TP)>=CODE "0" AND
PEEK (TP)<=CODE "9" THEN LET C$=
CHR$ (PEEK (TP)): GOTO 9915

9970 REM ■TERMINA COM ( PARA
          MATRIZES E FUNCOES
```

PRODUÇÃO DE NÚMEROS

Neste artigo do curso de lógica, projetamos um circuito conversor de sinais binários em dígitos decimais, utilizados nos visores de sete segmentos de calculadoras, relógios e alguns microcomputadores.

Em nosso projeto de circuitos, consideremos os nove algarismos decimais codificados em binários (DCB). Isto é, cada dígito decimal é armazenado num grupo de quatro dígitos binários, como mostra a tabela.

Decimal codificado em binário	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010 a 1111	Não utilizado

Visor de cristal líquido
As sete barras do LCD, que podem ser ligadas e desligadas individualmente, aqui designadas por letras de t a z.

Um visor com sete segmentos (ou barras) representa números decimais por meio de LEDs (Light-Emitting Diodes, "diodos emissores de luz") ou modificando a polaridade de determinadas barras de um LCD (Liquid Crystal Display, "visor de cristal líquido"). A ilustração mostra a denominação dos sete segmentos e as combinações para formar os dígitos de 0 a 9.

A ilustração da página seguinte permite determinar as barras ativadas a cada entrada no circuito. Por exemplo, se a entrada binária for 0100 (4, em notação decimal), o circuito ativará as barras designadas por u, v, y e z. Analogamente, a entrada 1000 (8, em notação decimal) fará com que todas as barras se iluminem. A tabela de validação para o circuito será

Decimal	Entradas				Saídas						
	A	B	C	D	t	u	v	w	x	y	z
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
	1	0	1	0	X	X	X	X	X	X	X
	1	0	1	1	X	X	X	X	X	X	X
	1	1	0	0	X	X	X	X	X	X	X
	1	1	0	1	X	X	X	X	X	X	X
	1	1	1	0	X	X	X	X	X	X	X
	1	1	1	1	X	X	X	X	X	X	X

Como não há um método fácil de analisar essa tabela, devemos simplificar as saídas separadamente, construindo, para cada uma, um mapa de Karnaugh que inclua as condições "ignore" (X). Em alguns casos, estas oferecem maior simplificação. Os sete mapas-k, um para cada linha, encontram-se na página seguinte. Delimitando os grupos por meio de círculos, produzimos uma expressão simplificada para cada linha de saída. Em alguns casos, fatoramos as expressões para simplificá-las ainda mais.

O primeiro mapa-k, por exemplo, lida com as entradas que ativam a barra t (essa barra é usada em todos os números exceto em dois). Podemos agora desenhar o circuito final.

Para t

X	X	1	1
X	X	1	1
1	1		1
X	X	1	

$t = C + A + B.D + B.D$

Para u

X	X	1	
X	X	1	1
1	1	1	1
X	X		1

$u = B + C.D + C.D$

Para v

X	X	1	1
X	X	1	
1	1	1	1
X	X	1	1

$v = D + \overline{C} + B$

Códigos de barras
A partir destes diagramas determinam-se as barras que devem ser ativadas para formar cada dígito. O algarismo 1 precisa somente das barras designadas por u e v. A tabela de validação na página anterior fornece as saídas necessárias para todos os dígitos.

Para w

X	X		1
X	X	1	1
1	1		1
X	X	1	

$w = A + C.D + B.C + B.D + B.C.D.$

Para x

X	X		1
X	X		1
1			1
X	X		

$x = C.D + A.D + B.D$
$= (A + \overline{B} + C).\overline{D}$

Para y

X	X		1
X	X		
1	1		1
X	X	1	1

$y = A + B.C + B.\overline{D} + \overline{C}.\overline{D}$
$= A + B.(\overline{C} + \overline{D}) + \overline{C}.\overline{D}$

Para z

X	X		1
X	X	1	1
1	1		
X	X	1	1

$z = A + B.C + B.C + B.D$
$= A + B.C + B.(C + D)$

O circuito projetado opera uma única célula. Como a maioria dos circuitos desse tipo tem oito ou mesmo dez unidades, poderia parecer necessário duplicá-lo para cada célula. No entanto, é possível partilhar um conversor entre todos os visores, num processo de multiplexação, que liga e desliga as unidades no visor em seqüência — a cada instante, apenas uma unidade admite informações vindas do conversor. Como isso acontece a alta velocidade, o visor parece estável.

UNITRON

Como a maioria das indústrias nacionais de microcomputadores, a Unitron surgiu do entusiasmo e da experimentação de estudantes de engenharia que não hesitaram em passar da teoria à prática.

Os drives elegantes
A Unitron foi a primeira indústria nacional a fabricar um slim drive — o Disk Drive UD 5, uma unidade de discos flexíveis com a metade da altura dos drives comuns

O ano de 1974 foi crucial para Geraldo Antunes e Ricardo Diez. Nessa época, os dois colegas de faculdade inauguraram uma fabriqueta onde produziriam um aparelho desenvolvido no curso de engenharia. Tratava-se de um dispositivo para acender lâmpadas fluorescentes, ativado por bateria. A produção inicial logo foi absorvida para uso em carrinhos de cachorro-quente e barracas de camping.

Um ano depois, os resultados revelaram-se tão satisfatórios que os dois amigos convidaram Roberto Petrone para fazer parte da sociedade e dirigir a transformação da oficina numa indústria capaz de produzir em escala.

No entanto, devido ao capital inicial mínimo, não restou aos sócios outra alternativa além da exploração das áreas menos visadas no mercado de componentes eletrônicos. Insistiram na mesma linha de produtos: a nova indústria estreou com o lançamento do Reatron — um conversor de corrente contínua, também para lâmpadas fluorescentes. Em 1976, a Unitron desenvolveu seu segundo produto, o Unilamp, um sistema automático para iluminação de emergência. Seis anos depois, os dois produtos haviam conquistado a confiança dos usuários, chegando a empresa a fornecer seus equipamentos para os metrôs de São Paulo e do Rio de Janeiro, e para a Rede Ferroviária Federal.

Teclado inteligente
Programado com memória especial, o teclado desse micro — dotado dos mesmos recursos do AP II — permite a digitação de maiúsculas e minúsculas, além da cedilha e de todos os acentos do português, como numa máquina de escrever

Ainda em 1982, a empresa instalou-se em prédio próprio e admitiu dois novos sócios, Sérgio Sá Moreira de Oliveira e Vilmar Gaertner. O primeiro foi o responsável pela introdução de novos sistemas e técnicas industriais; Gaertner teve, como diretor técnico, papel fundamental no projeto mais ambicioso da Unitron: entrar no setor da informática.

Nesse mesmo ano, os cinco sócios fundaram a Unitron Eletrônica para fabricar microcomputadores, dividindo instalações com a Unitron Indústria e Comércio. Até o final de 1984, a nova empresa colocou no mercado cerca de 6.500 microcomputadores, divididos entre os dois produtos básicos, o Unitron AP II e o Unitron T.I. (teclado inteligente).

O AP II, lançado em 1982, na Feira de Informática, é um microcomputador pessoal compatível com o Apple II, com 48 Kbytes de RAM e 12 Kbytes de ROM. Já o T.I. foi apresentado ao público em 1984, durante o Micro Festival, em São Paulo. Tratava-se de uma variação do micro original com caracteres da língua portuguesa e teclado programável.

A configuração comum aos dois equipamentos — um monitor de vídeo e unidade de discos flexíveis — possibilita a utilização de impressora e editor de texto. Assim, executam aplicações do tipo folha de pagamento ou de consulta a arquivos. Segundo a Unitron, as aplicações que mais atraem o público em geral estão na área de comunicações e acesso a informações. Por isso, o fabricante instalou em seus equipamentos a interface RS232C, que permite conexão a outro micro ou às redes públicas de informação, como Cirandão, Videotexto e Aruanda.

Uma das grandes preocupações da Unitron diz respeito ao aperfeiçoamento dos equipamentos existentes e ao desenvolvimento de novos produtos. Seguindo a tendência do mercado à verticalização, o fabricante estreou em 1985 sua linha de periféricos.

O primeiro aparelho, uma unidade de disquetes tipo slim, o Disk Drive UD 5, tinha metade da altura normal e constituía uma novidade no mercado nacional de drives. Esse produto foi projetado para atingir gradativamente 100% de nacionalização, graças à fabricação própria da cabeça de leitura/gravação e dos motores de acionamento. O Disk Drive UD 5 trabalha com face simples, densidade dupla e capacidade para armazenar até 20 Kbytes.

Como parte de sua estratégia de expansão, a Unitron Eletrônica desenvolveu uma política de comercialização de seus produtos também no exterior, principalmente em países latino-americanos, como a Argentina e o Suriname.

PRODUTORES DE MELODIA

Processados em bancos de filtros sob o controle de um micro, ruídos sem qualquer atrativo ganham ressonância e modulação, transformando-se em sons inauditos — os elementos da música do futuro.

O parentesco entre a música e a matemática vem de longe. Já no século VI a.C., o filósofo grego Pitágoras ficou intrigado ao ouvir, numa oficina de ferreiros, os martelos soarem tão "afinados" quando golpeavam as bigornas. Pesando vários deles, descobriu que um martelo com metade do peso de outro produzia um som com exatamente o dobro da freqüência — uma oitava acima — produzida por este. Estava descoberta a primeira das leis que governam as relações entre as alturas dos sons.

Na Idade Média, as catedrais ressoavam com os acordes de missas e motetos (composições corais polifônicas), cujo ritmo era tão preciso quanto suas próprias linhas arquitetônicas.

Muitas vezes, essa estrutura melódica alcançava tal complexidade que, segundo se acreditava, apenas os ouvidos de Deus podiam apreciar as relações numéricas contidas nas partituras, enquanto os simples mortais somente ouviam a música. Em vista desse caráter matemático, nada mais natural que o encontro dos universos da música e da computação.

Um desenvolvimento tecnológico recente que vem atraindo muita atenção é a MIDI (Musical Instrument Digital Interface, "interface digital para instrumentos musicais"), dispositivo que permite a intercomunicação de instrumentos eletrônicos (sintetizadores, computadores rítmicos e digital delays) e seu controle por micros pessoais ou dedicados (digital sequencers). Como a maioria dos instrumentos desse tipo hoje produzidos segue o protocolo MIDI, um novo horizonte se abre aos usuários de micros entusiastas de música.

A MIDI, contudo, não faz milagres. Não transformará, da noite para o dia, um operador de micro num Vivaldi ou num Egberto Gismonti. Os resultados sempre dependerão da habilidade musical e da imaginação de cada um, seja a música executada num sistema de sintetizadores ou num violão.

A música eletrônica é mais antiga do que se imagina. Bem antes da Segunda Guerra Mundial (1939-1945) alguns músicos já faziam experiências com geradores de ondas senoidais. Tais dispositivos elétricos provocavam vibração numa lâmina metálica, produzindo um som constante, cuja altura podia ser variada. Esse efeito sonoro foi muito usado nos filmes de ficção científica na década de 50. Por outro lado, os primeiros órgãos Hammond, lançados por volta de 1930, eram eletromecânicos e produziam vários tipos de timbres por meio da síntese aditiva — isto é, da sobreposição de senóides, gerando formas de ondas mais complexas.

Foi o rápido desenvolvimento da eletrônica durante a Segunda Guerra Mundial — em especial o surgimento da gravação magnética, na Alemanha — que proporcionou aos músicos ferramentas para a criação e manipulação dos sons de forma inédita. Eles começaram encomendando fitas com sons previamente gravados, musicais ou não. Esses minúsculos fragmentos

Partitura eletrônica
A "nova música" exigiu uma notação musical diferenciada. Esta partitura, por exemplo, do compositor alemão Karlheinz Stockhausen, foi elaborada de modo a lembrar circuitos eletrônicos. Ela contém representações de sons, compassos, sincronizações e até dá a posição dos microfones.

Impulso inicial

A música eletrônica não é tão nova quanto parece. O compositor britânico Ron Grainer (*sentado*) foi autor da primeira trilha eletrônica a conseguir algum sucesso — o tema do seriado *Doctor Who*, transmitido pela BBC para toda a Europa, no início dos anos 60

de fitas eram então meticulosamente combinados de modo a formar seqüências de eventos sonoros. A "nova música", como veio a ser conhecida, rompeu todas as regras da teoria musical ortodoxa e, ao mesmo tempo que fascinava alguns ouvintes, torturava muitos outros.

Os dispositivos para variar e distorcer os sons elementares produzidos nos primeiros osciladores, bem como para filtrar e modelar o resultado, foram se tornando mais controláveis, além de disponíveis em maior quantidade. Durante a década de 50, compositores como Karlheinz Stockhausen, na Alemanha, se empenhavam nos pequenos estúdios das emissoras de rádio, criando música eletrônica "pura". Em Paris, trabalhando com engenheiros de som da ORTF (Office de Radiodiffusion et Télévision Française), Pierre Schaeffer foi o pioneiro no que batizou de "música concreta", uma montagem musical utilizando sons do dia-a-dia.

Nos Estados Unidos, a Bell Telephone Laboratories construiu aquilo que veio a ser, provavelmente, o primeiro sintetizador. Ocupava várias salas e tinha por finalidade básica o estudo da síntese da voz humana. A empresa sabia que suas telefonistas com freqüência se enganavam por causa dos diferentes sotaques dos usuários. Isso resultava em elevado número de ligações erradas. Os especialistas — talvez muito otimistas para a época — esperavam resolver o problema com um sistema de respostas automático que usasse um sintetizador de voz universal.

Nessa época, graças à própria Bell, vários músicos americanos receberam seu batismo na eletrônica. Na Grã-Bretanha, trabalho similar estava sendo feito, se bem que em escala menos ambiciosa. Ainda assim, a BBC Radiophonic Workshop produziu um dos clássicos da música eletrônica, no início da década de 60 — o tema musical do seriado de televisão *Doctor Who*.

Software musical

O primeiro encontro entre a música e o computador ocorreu em 1957, quando Lejaren Hiller elaborou um programa para o computador Illiac, da Universidade de Illinois. As instruções eram decompostas em quatro agrupamentos de dados e depois transcritas em notação musical. Resultou uma obra para quarteto de cordas, em quatro movimentos, denominada *Illiac Suite*. A música em si, embora bem-arranjada — para celo, viola e dois violinos —, soava tortuosa e vaga. Mas não é difícil encontrarmos peças musicais da mesma época, produzidas convencionalmente, com sonoridade bem pior.

Poucos anos mais tarde, Hiller partiu para outra criação, dessa vez utilizando um computador IBM 7090. Ele fez um programa chamado Musicomp (Music Simulator-Interpreter for Compositional Procedures, "simulador-intérprete musical para procedimentos de composição"), que permitia maior flexibilidade de trabalho. E batizou a obra assim produzida — para um vocalista acompanhado por sons eletrônicos em fita — com o nome de *Cantata para computador*.

Seu trabalho foi apenas parte de um vasto esforço de pesquisas em universidades americanas nos anos seguintes. John Chowning, outro pioneiro, empregou o computador para estudar a percepção do som quando sua fonte muda de lugar. O uso que a Yamaha fez de seus trabalhos de pesquisa tem relação direta com os sintetizadores da série DX, fabricados em meados da década de 80 a partir do princípio de FM (Frequency Modulation, "freqüência modulada").

Com exceção das trilhas sonoras dos filmes de ficção científica, a música eletrônica permaneceu durante vários anos restrita ao campo da música clássica. O público só veio a conhecê-la melhor graças a mudanças de abordagem e de técnica por parte dos compositores de vanguarda. Os concertos dessa nova música, nos anos 60, apresentavam vários intérpretes tocando instrumentos convencionais e outros encarregados de processar eletronicamente estes sons, com unidades de separação de freqüências e filtros.

Desbravadores da música

Brian Eno tornou-se famoso em 1973, quando passou a usar seus sintetizadores em shows e gravações de astros como David Bowie e Robert Fripp. No Brasil, vários músicos, como Egberto Gismonti (o pioneiro), Wagner Tiso e César Camargo Mariano, utilizam sintetizadores em seus trabalhos, juntamente com instrumentos comuns.

Todos podiam estar acompanhando por uma partitura — que pouca semelhança tinha com a notação musical tradicional. Nelas havia, por exemplo, tanto indicações sobre posições de microfones e variações de filtros como anotações visuais relativas ao efeito sonoro a ser produzido. Em alguns casos, os músicos executavam por partituras que mais pareciam gráficos.

Pop eletrônico

Ainda na década de 60, músicos jovens começavam a passar mais tempo nos estúdios de gravação e a fazer experiências com a música eletrônica. Exemplo são os Beatles, que encontraram em George Martin um perito engenheiro de som e também um músico voltado para os avanços da música clássica de vanguarda. Ele encorajou o conjunto a usar todo o estúdio como instrumento musical, e logo os Beatles estavam trabalhando com técnicas de montagem de fitas e som sintetizado. O sucesso da música eletrônica junto ao grande público ocorreria, contudo, em 1967, com o lançamento da suíte *Switched on Bach,* de Walter Carlos — uma recriação eletrônica das obras de Bach realizada no sintetizador modular MOOG IIIp.

Alguns músicos fizeram nome utilizando determinados dispositivos de processamento de som. O guitarrista Jimmy Page, cujo estilo inspirava-se no dos músicos negros americanos dos anos 40 — só que usando uma série de controles de distorção —, produziu o som característico do Led Zeppelin. Esse som mais o uso que Jimi Hendrix fazia da realimentação e dos filtros de varredura rápida (os "pedais uá-uá") deram origem ao heavy metal.

Na década de 70, as várias unidades geradoras e processadoras de som (disponíveis desde os anos 50) incorporaram transistores em seus projetos. Elas se tornaram menores, leves e, portanto, não precisavam mais se restringir a um estúdio. Os guitarristas podiam fazer música ao vivo, criando inúmeros efeitos sonoros. Logo depois, pianistas e organistas também tiveram acesso aos sintetizadores e puderam levá-los para o palco.

Esses sintetizadores incluíam basicamente um conjunto de osciladores sintonizáveis, modeladores (com os quais se criam as características de irrupção, sustentação e decaimento do som), filtros variáveis, ring modulators (moduladores em anel, capazes de decompor os sinais em novas freqüências) e geradores de ruído.

O equipamento dos estúdios de gravação também se tornava mais sofisticado à medida que os músicos recorriam a eles para obter, no processo de produção, algo que não conseguiam criar no palco. A mesa de mixagem, destinada a combinar gravações em fita magnética de dezesseis ou 24 pistas, era grande demais para ser transportada com facilidade, e muitas das unidades de processamento exigiam longo tempo de montagem. Enquanto isso, nos Estados Unidos e na Grã-Bretanha, estava emergindo uma nova geração de produtores. Não raro, iniciavam como engenheiros e tinham mais familiaridade com o equipamento do que os músicos, de quem recebiam para produzir o "som certo".

Mago da eletrônica

Steve Levine é um consagrado produtor europeu de discos. Seu trabalho ficou conhecido pela excelente estrutura harmônica, que ele obtém combinando vozes, música eletrônica e sons de bateria produzidos pelo sintetizador Linn. Levine desenvolveu técnicas refinadas para os sistemas de gravação digital, criando obras de alta qualidade artística.

Na Jamaica, os engenheiros começaram a usar a mesa de mixagem como instrumento. Músicas inteiras, gravadas em fitas de várias pistas, eram desmontadas. A contribuição vocal e instrumental virou matéria-prima, resultando no pesado estilo dub, dependente da reverberação e das unidades de atraso de sinal.

O advento dos sintetizadores digitais trouxe a possibilidade de codificar sons não eletrônicos — processo conhecido como "amostragem". Os sintetizadores que reproduzem o som das baterias, como o Linn, tornaram-se equipamentos de estúdio muito procurados, logo passando a fazer parte de apresentações ao vivo. Em meados da década de 80, a amostragem e a manipulação de sons representavam a ponta-de-lança da criação musical.

Estúdios modernos e instalações de palco contêm, em geral, mais unidades de equipamento digital do que instrumentos analógicos. Alguns grupos bem-sucedidos combinam suas habilidades musicais às técnicas digitais de produção — como as utilizadas pelo produtor britânico Steve Levine, que trabalha com instrumentos e unidades de processamento de custo milionário.

A necessidade de interface para agrupar todos esses equipamentos e instrumentos (juntando-lhes o sistema operacional e a memória de um microcomputador) acabou reunindo os fabricantes de instrumentos musicais: em abril de 1983, eles apresentaram a primeira especificação da MIDI. A partir daí, poucas empresas ousaram anunciar um sintetizador não compatível com essa interface.

GRAUS DE PRECISÃO

As funções trigonométricas de seno e co-seno podem ser utilizadas nos programas em BASIC. Mas convém testá-las quando se trabalha com números muito pequenos ou muito grandes, para evitar erros.

Como a linguagem BASIC inclui as funções SIN (seno) e COS (co-seno), deve ser fácil calcular a posição de determinado ponto numa linha depois de girá-la certo ângulo. O co-seno de θ dá a posição no eixo x (a coordenada x), e o seno de θ, a posição no eixo y. Mas é preciso lembrar que a maioria das versões do BASIC trabalha com radianos em vez de graus. Assim, o primeiro passo consiste em estabelecer a diferença entre graus e radianos.

Chama-se "arco" qualquer parte de uma circunferência. Quando o arco tem o comprimento do raio, o ângulo assim determinado *(ver figura)* recebe o nome de "radiano". Se o raio for igual a 1, o arco terá o comprimento de uma unidade. Como existem 360 graus numa circunferência e seu comprimento é dado pela fórmula $2\pi r$, facilmente relacionamos graus e radianos:

Medida do radiano

$360° = 2\pi$ radianos;
$180° = \pi$ radianos;
$1° = \pi/180$ radianos $= 0{,}0174$ radiano.

Para achar o co-seno de um ângulo medido em graus, num programa BASIC, devemos primeiro converter a medida para radianos e, em seguida, usar a função COS. Experimente este pequeno programa escrito para o CP 500:

```
10 INPUT "ENTRE O ANGULO EM GRAUS"; A
20 LET B# = A * 0,0174
30 LET C# = COS(B#)
40 PRINT "O CO-SENO DE";A;"GRAUS E";C#
50 END
```

O símbolo # após o nome de variáveis indica que elas armazenam dados com dupla precisão. Empregamos tal notação nas versões do BASIC do TRS-80 e compatíveis — CP 300, CP 500, Digitus, Sysdata etc.

Modificando o programa pela introdução da função seno, podemos construir uma tabela com todos os valores de θ, de 0 a 360 graus. Marcando esses valores no eixo y de um gráfico (o eixo x representa os valores de θ em radianos), o resultado será a curva do seno, o desenho dos pontos de interseção da hipotenusa com a circunferência de raio unitário no eixo y para todos os ângulos de rotação. Trata-se de um modo alternativo de descrever matematicamente uma circunferência.

Talvez a programação em BASIC de seu micro permita que as funções SIN e COS funcionem tanto em graus quanto em radianos, mas em geral isso não acontece. Se você preferir trabalhar com graus, poderá criar uma "função definida pelo usuário" que faça as conversões. Eis um programa que usa esse recurso:

```
10 REM UMA FUNCAO DEFINIDA PARA
   TRABALHAR COM GRAUS
20 DEF FNGRASEN(D#) = SIN(D# * 0,017453293)
30 INPUT "ENTRE O ANGULO EM GRAUS";D#
40 PRINT "O SENO DE"#;D#;"GRAUS
   E" FNGRASEN(D#)
50 END
```

A linha 20 define a função GRASEN (graus/seno), cujo único parâmetro é a variável de dupla precisão D#. A metade direita da definição mostra como calcular o valor a ser devolvido pela função — o seno de determinado ângulo medido em graus. Para chamar uma função definida pelo usuário, usamos o nome normal da função com o valor que será operado entre parênteses. Observe, no entanto, que se executa a linha contendo a definição antes que se possa chamar a função.

Números, pequenos e grandes

Um dos problemas quanto ao uso da função seno na linguagem BASIC é que nem todas as suas versões conseguem manipular a função corretamente quando o valor do ângulo se aproxima de zero. Nesse caso, o valor do seno de θ também se aproxima de zero: o seno de θ vale zero quando θ é igual a zero.

Em outras palavras, quando o ângulo tende a zero, o mesmo acontece com o arco de circunferência que define θ, enquanto o ponto em que a hipotenusa intercepta o círculo se aproxima cada vez mais de zero no eixo y.

O BASIC é limitado, não conseguindo trabalhar com valores muito grandes ou muito pequenos além de determinados limites. Se θ for muito pequeno — por exemplo, igual a 1,0E-36, ou seja, 1 x 10 elevado à potência (−36) —, o BASIC não conseguirá processá-lo e simplesmente devolverá um valor igual a zero para o seno desse número.

Por esse motivo, antes de usar a função seno, teste o BASIC de seu micro com o pequeno programa que segue, também escrito para o CP 500. O resultado é confiável até que θ atinja um valor menor que 1,0E-38 (37 zeros após a vírgula).

DUPLA DINÂMICA

Abandonando as limitações até agora impostas a nossos arquivos, veremos conceitos como desalocação, indicadores, stacks e heaps, que permitem o processamento de arquivos de tamanho não especificado.

Dois procedimentos padronizados (NEW e DISPOSE) e um tipo especial de dados (POINTER) proporcionam ao PASCAL uma extraordinária flexibilidade na "alocação dinâmica" e na desalocação de memória. Um tipo POINTER é apenas uma variável que, em vez de conter um dado como seu valor — por exemplo, um valor integer —, indica ou "aponta" um valor integer ou qualquer objeto de dados, estruturado ou não. Para definir um tipo POINTER, usamos uma seta apontada para cima (↑) ou o sinal de acento circunflexo ^ antes do identificador (ou do tipo de dado que desejamos indicar):

```
TYPE
  MATRIZ = ARRAY[1 .. 100] OF REAL;
  INDIRETO = ^MATRIZ;
```

Todos os compiladores PASCAL com padrão ISO também aceitam o símbolo alternativo @. A definição do tipo reserva uma única posição de memória, que será usada para conter o endereço de uma matriz somente após ter ela sido criada pelo procedimento NEW. Enquanto isso, o valor de POINTER permanece indefinido, como ocorreria com qualquer outra variável, de modo que

```
VAR
  ENDERECO : INDIRETO;
  NUMERO : INTEGER;
```

reservará espaço para um endereço na máquina (16 bits num micro de 8 bits) e para um valor integer, sem que sejam inicializados com qualquer valor específico.

Da mesma forma que poderíamos inicializar o valor integer em zero antes de usá-lo, o indicador pode receber o valor especial NIL. Essa palavra reservada do PASCAL significa apenas que o POINTER não está apontando nada, equivalendo ao valor numérico zero (que implica ausência de um número efetivo). As duas variáveis do exemplo receberiam a seguinte atribuição:

```
ENDERECO := NIL;
NUMERO := 0;
```

Exemplificando o emprego de NEW e DISPOSE, usaremos, para simplificar, somente valores integer. Quando se faz necessária uma variável apontadora para indicar um novo item, chamamos o procedimento NEW, que reserva espaço para o item e coloca seu endereço no POINTER. A seguir, o item de dados é referenciado por meio da "referenciação" ao POINTER (P) com a notação P↑ ou P^. Observe que os sinais agora vêm após o identificador do POINTER, podendo ser considerados como "os itens para os quais p está apontando". Não faz sentido referenciar um POINTER indefinido ou que não aponte para algo.

Desalocação dinâmica

Após utilizar os dados criados pelo uso do procedimento NEW, chamamos o procedimento DISPOSE. Trata-se da operação oposta a NEW, porque, com DISPOSE, a memória volta ao conjunto dinâmico, e o valor do POINTER se torna indefinido. O programa DOISMAISDOIS ilustra esses pontos:

```
PROGRAM DOISMAISDOIS (OUTPUT);
TYPE
    POINTER = ^INTEGER;
VAR
    P1,
    P2^ : POINTER;
    RESPOSTA : INTEGER;
BEGIN
  NEW (P1);
  P1 = 2.
  NEW (P2);
  P2^ := P1^;
  RESPOSTA := P1^ + P2^;
  WRITELN (P1^, '+', P2^, '=', RESPOSTA);
  DISPOSE (P1);
  DISPOSE (P2);
END.
```

Com esse método pouco usual de adição, notamos o seguinte:

- As variáveis dinâmicas são anônimas — não há identificador de variável (como N) que comporte o valor 2 em qualquer ponto do programa; a referência a esses itens se faz indiretamente.

- Após o segundo DISPOSE, a única memória usada é um valor integer isolado (RESPOSTA); todos os dados dinâmicos deixaram de existir.

O endereçamento indireto implícito no uso do POINTER tem duas conseqüências. Em primeiro lugar, nunca sabemos (nem precisamos saber) quais os endereços efetivos. O único "endereço absoluto" disponível para o programador é NIL.

Do mesmo modo, estamos livres para usar, corrigir e, em seguida, reutilizar qualquer parte da memória disponível sem que seja preciso reorganizar os dados descartados. O PASCAL cuida do gerenciamento da memória, e a única informação de que precisaremos refere-se à quantidade de memória disponível. Para obtê-la, empregamos a função não padronizada MEMAVAIL ou, em alguns equipamentos, a função FREE, que fornece o número restante de bytes de memória.

Símbolos indicadores

——— ?
POINTER indefinido (após uma declaração, antes de uma atribuição ou após DISPOSE).

———|
POINTER que não aponta para nada (após atribuição NIL).

[|]
Módulo de registro com um campo de dados e um POINTER.

Outra extensão útil é a função SIZEOF (TYPEIDENTIFICADOR), que fornece o tamanho em bytes de um determinado tipo. A RAM do usuário se divide em estruturas internas — o "stack" e o "heap". Emprega-se o primeiro para chamadas a funções e procedimentos, armazenando seus endereços de retorno, dados locais e valores resultantes. Todos os dados dinâmicos são alocados no heap, que opera de forma semelhante ao stack. No entanto, o heap não é uma estrutura de dados LIFO (Last In First Out, "último a entrar, primeiro a sair") e inicia no lado oposto da memória, crescendo em direção ao stack. O uso ambicioso de NEW e/ou da recorrência pode levar a uma "colisão stack-heap", evitável por meio de

```
IF SIZEOF (OBJETO) > PORCENT * MEMAVAIL
   THEN .
```

sendo OBJETO o identificador de tipo dos dados que estão para ser criados e PORCENT um valor na região de 0,7, que reserva 30% da memória para o stack e 70% para o heap. Se — como é efetivamente o caso — MEMAVAIL/FREE funcionar como uma "marca de limite máximo", será possível manter uma "contagem de dados não em uso" dos itens descartados.

Estruturas ligadas

Percebemos a verdadeira força dos POINTERS quando criamos estruturas ligadas (linked) em forma de árvore, listas vinculadas individual ou duplamente, estruturas circulares etc. No caso de strings de caracteres, podemos — e freqüentemente o fazemos — utilizar uma matriz com um tamanho fixo de, digamos, oitenta caracteres. Se tivermos uma matriz de strings para armazenar um documento, por exemplo, cada linha em branco ainda ocupará 80 bytes de armazenamento. Do mesmo modo, não podemos representar linhas com mais de oitenta caracteres — toda a estrutura de dados é rígida demais.

Um compilador PASCAL tem de distinguir entre identificadores de qualquer tamanho. Assim, por exemplo, como representar com precisão a variação de tamanho característica do mundo real? A estrutura mais natural seria um registro contendo dois campos: um para cada item de dados (CHAR, nesse caso), e um segundo campo POINTER apontando o registro seguinte, se houver, na lista:

```
TYPE
   STRING = ^CARACTERE;
   CARACTERE = RECORD
                  CH : CHAR;
                  PROX : STRING
                  END;
VAR
   LINHA : STRING;
BEGIN
  LINHA := NIL;
```

Representa-se um string vazio pela atribuição do valor NIL. Qualquer outra seqüência exigirá um novo registro contendo cada CHAR e um outro POINTER para indicar o registro seguinte. O último registro terá seu próximo campo inicializado em NIL para que se possa detectar o final do string. Um procedimento para imprimir o string seria:

```
WHILE LINHA < > NIL DO
  BEGIN
    WRITE (LINHA ^. CH);
    LINHA := LINHA ^. PROX
  END
```

Observe a recorrência da estrutura, definida em termos de si mesma. Não se pode atribuir ao campo indicador um tipo que tenha sido definido por completo; assim, faz-se uma referência "antecipada".

Listas ligadas

VAR
LINHA : STRING; ?

O PASCAL reserva espaço para um endereço (não definido) na memória conforme definição do tipo.

LINHA := NIL;

Um POINTER pode ser inicializado em "zero" atribuindo-se o valor especial NIL.

NEW (LINHA); ? ?

LINHA CH PROX

O procedimento NEW reserva espaço para dados na memória e armazena o endereço da posição reservada na variável POINTER.

LINHA ^. CH := 'a'; 'a' ?

A referência a itens de dados não está explícita, sendo feita por "endereçamento indireto" com a notação ^ ou ↑.

LINHA ^. PROX := NIL; 'a'

P := LINHA;

```
WITH P ^ DO
  BEGIN
    NEW (PROX);
    PROX ^ CH := 'b'
  END
```

'a' 'b' ?

Lista de ligação simples

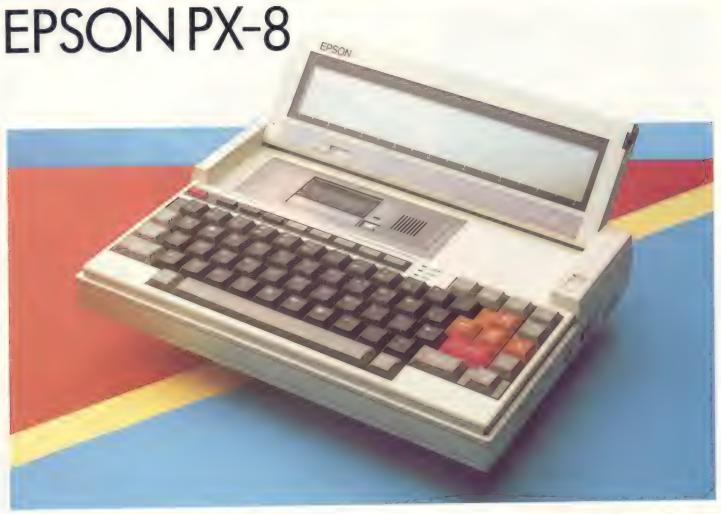
De modo sucessivo
O PASCAL permite ao programador criar "listas ligadas" — poderosas estruturas de dados nas quais cada item aponta o item seguinte da lista. Esta pode ser de ligação simples (entre itens sucessivos), de ligação dupla (cada elemento possui POINTERS tanto para o item seguinte como para o anterior) ou circular (onde o último item, numa lista de ligação simples, aponta o primeiro item).

Programa ListaCirc

```
PROGRAM LISTACIRC (INPUT, OUTPUT)
        PROPOSITO  INSERIR REGISTROS CONTENDO DADOS
         DE VARIOS TIPOS EM UMA LISTA CIRCULAR
         DINAMICAMENTE ALOCADA  OS DADOS SAO
         COLOCADOS EM ORDEM DE UM CAMPO CHAVE
         (NOME) ALFABETICAMENTE  TORNANDO
         DESNECESSARIOS ALGORITMOS DE CLASSIFICACAO

CONST
    MAXSTRING   =  25
    ESPACO      =

TYPE
    CARDINAL    =  0   MAXINT
    TAMSTRING   =  0   MAXSTRING,
    STRING      =  PACKED ARRAY [TAMSTRING] OF CHAR,
    REGISTRO    =  RECORD
                      NAME   : STRING,
                      (OUTROS CAMPOS)
                      DEBITO   : CARDINAL
                   END


    NO          =  RECORD
                      ITEM     REGISTRO
                      PROXIMO    POINTER
                   END
```

Linha circular
Este programa permite inserir registros contendo dados mistos em uma lista circular dinamicamente alocada.
Os dados são organizados por meio de uma chave alfabética (NOME), tornando assim desnecessários os algoritmos de classificação.

```
      POINTER       =     NO
   VAR
      LISTA              :   POINTER;

      11111111111111111111111111111111111111
   PROCEDURE LERREG (TOPO   POINTER);

      VAR
             DADO   :   REGISTRO.
             OK     :   BOOLEAN.

         22222222222222222222222222222222

   PROCEDURE SALTABR (VAR F · TEXT);
      IGNORA BRANCOS
    VAR
       FEITO          :   BOOLEAN;
    BEGIN
      FEITO           : = EOF (F);


      WHILE NOT FEITO DO
        BEGIN
          FEITO := (F ˆ> ESPACO) OR EOLN (F);

           IF NOT FEITO THEN
             BEGIN
               GET (F);
               FEITO   =  EOF (F);
             END.
        END

   END; SALTABR
      22222222222222222222222222222222

 PROCEDURE LERCARD (VAR F : TEXT;
                        VAR N   CARDINAL;
                        VAR OK   BOOLEAN).
     OBTEM UM NUMERO POSITIVO·
  TYPE
        NUMERO = 0 . 9;
  VAR
        DIGITOS  :   SET OF CHAR;
        FEITO    :   BOOLEAN

        333333333333333333333333

 FUNCTION VALOR (DIGITO   CHAR)   NUMERO,
    CONVERTE CHAR PARA VALOR NUMERICO
   BEGIN
     VALOR := ORD(DIGITO) - ORD('0');

  END; :VALOR,

        333333333333333333333333

 FUNCTION LEGAL (N     CARDINAL,
                  DIGITO · CHAR)         BOOLEAN;
      PROXIMIDADE A OVERFLOW DE CARDINAL,
   BEGIN
     IF N = MAXINT DIV 10
       THEN
         LEGAL := VALOR(DIGITO) < = MAXINT MOD 10
       ELSE
         LEGAL := N < MAXINT DIV 10

   END; ;LEGAL;
       3333333333333322222222222222

 BEGIN      ;LERCARD;

   DIGITOS ·= ['0'   '9'];
   OK          := NOT EOF (F);

   IF OK THEN ,E UM DIGITO?;
     OK := F; IN DIGITOS;

   N ·= 0;
   FEITO := NOT OK,

   WHILE OK AND NOT FEITO DO
     BEGIN ,N PARA A BASE 10 ·
       N   =  10 * N + VALOR (F,);
       GET (F),
       FEITO := EOF (F),

       IF NOT FEITO THEN
         FEITO := NOT (F ˆ IN DIGITOS),

       IF NOT FEITO THEN
         OK := LEGAL (N, F ˆ)
   END.

 IF NOT EOF (F) THEN
   READLN (F)

 END; ;LERCARD;
     22222222222222222222222222222222!

 PROCEDURE LERLINHA (VAR S   STRING);
    :LE UMA LINHA DO TECLADO;
  VAR
        INDICE   0   MAXSTRING
        SIMBOLO   CHAR
 BEGIN
   SALTABR (INPUT);
   INDICE  .= 0;

   WHILE NOT EOLN AND (INDICE  <  MAXSTRING) DO
     BEGIN ;INSERE 1 CHAR NO STRING,
```

```
             INDICE   = SUCC (INDICE);
             READ (SIMBOLO);
             S[INDICE] := SIMBOLO;
           END.

       IF INDICE < MAXSTRING
         THEN ;COMPLETA COM NULOS;
           FOR INDICE ˆ= INDICE + 1 TO MAXSTRING DO
             S[INDICE] := CHR (0)
         ELSE
           IF NOT EOLN THEN ,EXCESSO DE CHARS;
             WRITELN ('ATENCAO · TRUNCADO').

       READLN;

    END; ;LERLINHA;
      '22222222222222222222222222222222

  PROCEDURE INSERE (LISTA   POINTER
                     DADO   REGISTRO)
    VAR
         PPROX
         LIGA   POINTER
         ALFA · STRING;
   BEGIN ;COLOCA DADO NO TOPO DA LISTA;
     LISTA ˆ ITEM := DADO.
     PPROX   = LISTA
     ALFA    = PPROX ˆ PROXIMO ˆ ITEM  NOME

     WHILE DADO  NOME  ·  ALFA DO
       BEGIN  PESQUISA NA LISTA
         PPROX   = PPROX ˆ PROXIMO
         ALFA   = PPROX   PROXIMO ˆ ITEM  NOME
       END

           ;SEM VERIFICACAO DE DUPLICIDADE;
    NEW (LIGA). ;CRIA UM NO;
    LIGA ˆ ITEM := DADO; ;INSERE DADOS;
    LIGA ˆ. PROXIMO := PPROX ˆ PROXIMO; ;E LIGA;
    PPROX ˆ. PROXIMO := LIGA ;A LISTA·

   END; ,INSERE;
     ;222222222222222222211111111111111111!

 BEGIN ;LERREG;

   WRITE ('NOME? = > ');
   LERLINHA (DADO . NOME);
         ,PARA NUMA ENTRADA NULA;
   WHILE DADO . NOME [1] < > CHR (0) DO
     BEGIN
       REPEAT
         REPEAT
           WRITE ('DEBITO?        15)

           IF EOLN (INPUT) THEN
             READLN (INPUT)

           SALTABR (INPUT)

         UNTIL NOT EOLN (INPUT)

         LERCARD (INPUT, DADO . DEBITO, OK);

         IF NOT OK THEN ;SEM DIGITOS
           WRITELN ('*** ERRO -   . 20.
             'REENTRE ***');

       UNTIL OK


       INSERE (TOPO, DADO),
       WRITELN ('(RETURN SE ACABOU)' : 40);
       WRITE ('NOME?');
       READLN (DADO . NOME);
     END

   END; ;LERREG;
     1111111111111111111111111111111111111

  PROCEDURE DISPLAY (LISTA : POINTER);

    VAR

           MOSTRA   POINTER
    BEGIN  IMPRIME O TOPO DA LISTA
      MOSTRA   = LISTA   NEXT
      PAGE (OUTPUT)
      WRITELN (LISTA ORDENADA   )
      WRITELN,
           ;EXISTEM MAIS ITENS A MOSTRAR?;
      WHILE NOT (MOSTRA = LISTA) DO
        BEGIN
          WITH MOSTRA ˆ ITEM DO
            WRITELN (DEBITO : 8,NOME : 30);

          WRITELN
          MOSTRA      MOSTRA   PROXIMO
        END

   END; ;DISPLAY;
     1111111111111111111111111111111111111

 BEGIN ;LISTACIRC - PROGRAMA PRINCIPAL;

   NEW (LISTA)  CRIA NO VAZIO NO TOPO;
   LISTA   PROXIMO := LISTA; ;QUE SE APONTA;
   PAGE (OUTPUT);
   WRITELN ('ENTRE COM OS DADOS .        SOBRENOME'
          'ANTES, DEPOIS COM O DEBITO ·');
   WRITELN,
   LERREG (LISTA); ;COLOCADOS NA LISTA;
   DISPLAY (LISTA); ;MOSTRA EM ORDEM;
   WRITELN ('——— ENCERRADO ———' : 40)
 END
```

EPSON PX-8

Alimentado por bateria, o Epson PX-8 possui RAM de 64 Kbytes, CP/M e diversos softwares gravados em chips de ROM, que o convertem num dos mais competitivos micros portáteis do mercado internacional.

Acondicionado num estojo do tamanho de uma lista telefônica, o PX-8 não se parece com um computador, graças a suas reduzidas dimensões e seu pequeno peso (cerca de 2,3 kg). No entanto, deslizando-se a tampa corrediça do estojo, surgem um teclado completo e um visor de cristal líquido (LCD, Liquid Crystal Display), dobrado, que permite onze inclinações diferentes, conforme o ângulo mais adequado ao operador. Sob o visor, encontra-se o gravador de microfita cassete.

O teclado, tipo máquina de escrever, apresenta 72 teclas, codificadas em cores para indicar suas finalidades. As alfanuméricas, em marrom-escuro, estão distribuídas no formato padrão QWERTY, nas versões americana e inglesa. Nesta, o caractere $ substitui o # (hash mark, "cancelamento") encontrado no modelo americano. Todos os caracteres de outras línguas podem ser produzidos pelo teclado, com o simples acionamento de algumas chaves.

Quatro teclas laranja controlam os movimentos do cursor. Há também as teclas [Insert], [Delete] e [Home], e três outras que acionam funções do sistema ([Escape], [Pausé] e [Help]), além de cinco para funções programáveis pelo usuário.

A operação do PX-8 é fácil com o aparelho colocado no colo, devido a seu design compacto. Torna-se menos funcional, no entanto, quando instalado sobre uma mesa, pois as teclas requerem pressão direta de cima para baixo. Duas pernas retráteis acopladas proporcionam certa inclinação à unidade, mas não resolvem satisfatoriamente o problema. As teclas [Caps Lock], [Number] e [Insert] servem para mudar os modos de tela do PX-8 — três pequenas luzes vermelhas no painel indicam o modo em uso.

A documentação, bastante abrangente, consiste em dois volumosos manuais: um, destinado ao usuário, trata da instalação do aparelho, uso do hardware, do software e das operações CP/M. O outro é um guia de referência para programação em BASIC.

O manual do usuário inclui também mapas da memória, listas completas dos caracteres disponíveis e respectivos códigos, além de um programa em ASSEMBLY bastante longo, para gravar e carregar gráficos de tela em disquete. O guia de referência começa explicando como instalar e utilizar o BASIC (fornecido, assim como o pacote

Boa opção
O micro portátil PX-8 é fabricado pela Epson, famosa por suas impressoras matriciais. Esse equipamento tem 64 Kbytes de RAM, visor de cristal líquido, CP/M e vários pacotes de software residentes em chips de ROM.

Grande LCD
O visor de cristal líquido, de 80 x 8 caracteres, utiliza a bateria do micro e oferece resolução gráfica de 480 x 64 pixels.

Software em ROM
Retirando-se um pequeno painel da parte inferior do PX-8, aparecem os conectores onde se ligam os softwares gravados em EPROM. Aí está instalado o WordStar, assim como o sistema operacional CP/M. Para passar do WordStar ao Calc, basta trocar as EPROMs.

de software, num chip ROM). O outro volume discorre sobre a natureza da programação, num exame dos vários modos de tela do PX-8, além de explicar em detalhe todos os comandos BASIC disponíveis.

Por dentro do Epson

O PX-8 utiliza CPU de CMOS (Complementary Metal Oxide Semiconductors, "semicondutores complementares de óxido metálico") compatível com o Z80. Os chips CMOS exigem muito menos energia do que os comuns. Acrescido ao baixo consumo do visor, o emprego destes chips permite o funcionamento da unidade com fonte de alimentação recarregável.

Acompanham o micro duas baterias, sendo uma de reserva. Garantidas por até quatro anos, devem ser carregadas antes de o computador entrar em uso; assim, é preciso esperar oito horas a partir da instalação do equipamento até o início da operação. A carga é suficiente para quinze horas de operação contínua.

Logo que o PX-8 estiver pronto para funcionar, seu sistema operacional deve ser inicializado. Os passos necessários para isso incluem entrar com dia, data, hora, e cuidar de algumas tarefas de manutenção.

Uma delas é a formatação de um disco simulado em RAM. O PX-8 reserva uma porção da RAM — selecionável pelo usuário entre 9 e 24 Kbytes —, tratando-a como um disco de armazenamento (RAMDISK). O sistema operacional utiliza essa área da memória como se fosse um disco externo. Antes de usá-lo, formata-se o RAMDISK, especificando-se a quantidade de RAM que se quer utilizar. A Epson também oferece uma unidade de disco adicional em RAM, contendo 120 Kbytes extras.

O PX-8 possui na memória o sistema operacional CP/M, residente em ROM, e exibe no LCD, sob forma de menu, uma lista de utilitários em CP/M e um diretório do software residente em ROM.

Softwares

O software, em cassete, disquete ou ROM, é gravado em chips EPROM que se encaixam num conector sob o micro. Os softwares que acompanham o PX-8 — WordStar (processador de texto), Calc (planilha) e Scheduler (banco de dados), em versões portáteis — são fornecidos nesse formato de módulo, assim como o interpretador BASIC. Para selecionar determinado aplicativo, utilizam-se as teclas do cursor para indicar a escolha desejada e aperta-se [Return]. Carrega-se então o programa selecionado da ROM para a RAM.

O visor de cristal líquido, de 80 x 8 caracteres, apresenta resolução gráfica de 480 x 64 pixels. A maior desvantagem desse tipo de tela está na lentidão. Embora os caracteres apareçam com rapidez satisfatória, à medida que são digitados, qualquer tipo de correção — sobretudo quando envolve palavras ou sentenças completas — leva um tempo excessivo para ser realizado.

O software fornecido com o PX-8 é de fácil compreensão. A Epson oferece também um programa de telecomunicações, para uso com modem, e um programa destinado a transferir arquivos do PX-8 para computadores maiores, como o QX10, da Epson. Funcionando com CP/M, o PX-8 aceita outros softwares desenvolvidos nesse sistema.

O BASIC do PX-8 é uma versão Microsoft melhorada pela Epson. Inclui o comando AUTO para numerar e renumerar as linhas dos programas, um editor de tela completo, bem como comandos gráficos e sonoros, para comunicações através da interface RS232C incorporada, além de comandos que permitem usar o gravador microcassete como unidade de disco.

Em resumo, o Epson PX-8 é um microcomputador ideal para jornalistas, homens de negócios e todos que precisam de um equipamento pequeno mas eficiente, de fácil transporte em viagens e utilizável em qualquer local, mesmo os que não possuem rede elétrica.

Saída para alto-falante
Amplifica o som quando em conexão com um alto-falante externo.

Saída A/D
Liga dispositivos externos como voltímetros, medidores de temperatura ou balanças

Saída para leitora de código de barras
Permite o uso do PX-8 para controle de preços e de estoques.

Processador auxiliar 7508
Converte em digitais os sinais analógicos recebidos pelo conector A/D.

RAM
O PX-8 contém os 64 Kbytes de RAM exigidos pelo CP/M. A bateria de reserva assegura a manutenção do conteúdo da memória quando se desliga o aparelho.

CPU
É a versão CMOS do familiar microprocessador Z80.

Saída para impressora
Acopla o computador a uma impressora serial.

Saída para comunicações
Possibilita comunicação com outros computadores, seja diretamente, seja por intermédio de modem e rede telefônica.

Saída do bus interno (paralela)
Utilizada para o disco de RAM adicional com 120 Kbytes.

Controlador de periféricos
Permite comunicação com o alto-falante interno do computador e com periféricos externos, como impressora, unidade de disco ou gravador cassete.

Software em EPROM
Permite o uso de aplicativos em EPROM. Aqui está encaixado o WordStar, que pode ser substituído por outro módulo, como o Calc.

ROM de utilitários CP/M
O PX-8 pode utilizar grande quantidade de software disponível em CP/M.

EPSON PX-8

MICROPROCESSADOR

CPU, de CMOS, compatível com o Z80.

CLOCK

2.4 MHz

MEMÓRIA

64 Kbytes de RAM, 32 Kbytes de ROM, mais 6 Kbytes de memória de vídeo.

SISTEMA OPERACIONAL

CP/M.

VÍDEO

Visor de cristal líquido. Modo texto, 80 x 8 caracteres; modo gráfico, 480 x 64 pixels.

INTERFACES

Saída para impressora serial RS232C; saída para leitura de código de barra; entrada analógica.

LINGUAGENS

BASIC em versão Microsoft melhorada, operando em CP/M.

TECLADO

Estilo máquina de escrever QWERTY, 72 teclas, incluindo quatro para controle do cursor e cinco para funções programáveis.

DOCUMENTAÇÃO

Dois grandes volumes encadernados em espiral: um manual de operações e um guia de referência de BASIC.

TK! SOLVER

Examinamos aqui um programa elaborado pelos criadores do VisiCalc, que leva o conceito de planilhas eletrônicas a uma nova direção e a um novo alcance: o processamento de equações.

As planilhas eletrônicas mostram-se muito úteis em várias tarefas matemáticas executadas em microcomputadores. Para pessoas acostumadas a manipular grandes matrizes de linhas e colunas com lápis e calculadora, as planilhas eletrônicas poupam tempo e energia. Por outro lado, têm limitações significativas. O formato de linhas e colunas, ideal para contabilidade ou outros modelos financeiros, com freqüência é incômodo e às vezes inútil para aplicações matemáticas e científicas de nível mais alto. Além disso, as planilhas eletrônicas apresentam uma estrutura rígida demais para o trabalho com equações.

A Software Arts (companhia americana que criou a planilha VisiCalc, um sucesso de vendas) desenvolveu o programa TK!Solver, que supera as planilhas atualmente em uso, tanto na forma como na função. TK vem de tool kit (conjunto de ferramentas), e Solver (solucionador) refere-se à parte do programa que processa as equações. Além de diferir das planilhas comuns no formato da tela, o TK!Solver oferece algumas características exclusivas:

Resolução múltipla. As fórmulas das planilhas comuns resolvem equações de uma só variável. O TK!Solver determinará qualquer incógnita de uma equação, se forem fornecidos dados suficientes para isso.

Iteração. Quando um valor necessário à resolução de uma equação falta ou é desconhecido, pode-se entrar com um valor estimativo. O TK!Solver o usará como ponto de partida e resolverá a equação por uma série de aproximações sucessivas.

Conversão de unidades. O programa converte valores (pés em metros, dólares em libras etc.) instantaneamente, a partir de tabelas apropriadas.

Funções matemáticas. O TK!Solver tem grande número de funções incorporadas.

O programa TK!Solver funciona por meio de três planilhas interligadas, cada qual com uma função específica.

A planilha Variáveis contém os nomes de todas as variáveis definidas; colunas para os valores de entrada do usuário e para os de saída do programa; e espaço para indicar unidades associadas e para comentários sobre cada variável. Essa planilha aparece na parte superior da tela inicial do programa. Cada variável também é descrita em pormenores numa subplanilha Variáveis separada.

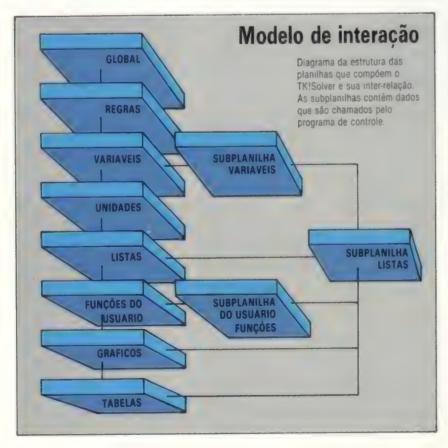
A planilha Regras é usada para introduzir as equações que o TK!Solver deve solucionar. Uma equação pode ter até duzentos caracteres de comprimento e obedecer às convenções matemáticas padronizadas para notação e operações. A planilha Regras ocupa a parte inferior da tela de abertura do TK!Solver.

A planilha Unidades armazena as informações necessárias para converter as unidades de medida associadas às variáveis do modelo.

O TK!Solver usa essas três planilhas para efetuar a maior parte das operações. Há outras, como a Global, em que o usuário pode adaptar, para usos específicos, alguns dos procedimentos operacionais do TK!Solver. A planilha Listas guarda uma matriz de valores para as variáveis; a Funções do Usuário executa funções por ele definidas; e há também aquelas para desenhar pontos (plotar) e imprimir tabelas de valores.

Um modelo muito simples, adaptado do manual do usuário do TK!Solver, calcula a milhagem (autonomia em milhas) e a velocidade média de uma viagem de automóvel, além de conver-

Modelo de interação

Diagrama da estrutura das planilhas que compõem o TK!Solver e sua inter-relação. As subplanilhas contêm dados que são chamados pelo programa de controle.

ter milhas em metros e seus múltiplos. Na tela de abertura do TK!Solver encontramos o cursor na planilha Regras, na parte inferior do vídeo. Começamos definindo as variáveis em equações adequadas teclando

Distância/Tempo = Velocidade

e pressionando [Return]. O TK!Solver transfere os nomes das variáveis das equações para a planilha Variáveis, na parte superior; examina a equação; e imprime as variáveis na coluna Nomes da planilha Variáveis, na mesma ordem em que aparecem na equação.

Um asterisco na coluna Status (estado) significa que a equação não foi satisfeita, pois não se introduziram valores em Variáveis. Entramos então com a segunda equação:

Distância/Combustível = Milhagem

Fornecida esta fórmula, todas as cinco variáveis definidas serão listadas na coluna Nomes da planilha Variáveis, conforme a *Tela 1*.

Tela 1: Equações e variáveis

Pressionando a tecla [;], movemos o cursor da planilha Regras para a Variáveis, onde podem ser introduzidos novos valores. O cursor surge na coluna de entrada ao lado da primeira variável, Distância. Digitamos então os valores no local adequado, conforme a tabela *à direita*, e acionamos [Return] ou a seta que aponta para baixo. Os valores Velocidade e Milhagem ficam em branco, pois cabe ao TK!Solver encontrá-los. Seus valores calculados serão apresentados na coluna Output (saída).

Para determinar a velocidade e a milhagem, pressionamos o ponto de exclamação [!], que o TK!Solver entende como tecla de "ação". Aparecerá a expressão Direct Solver acima da planilha Variáveis, indicando que o programa já recebeu os dados necessários para obter uma solução direta. Os valores Velocidade e Milhagem serão apresentados como saída. Apagamos então o que introduzimos anteriormente e obtemos um valor para Distância, fornecendo ao TK!Solver novos valores de velocidade e tempo, ou de milhagem e combustível.

Os valores introduzidos até agora não têm unidades associadas. Não podemos simplesmente digitar milhas, ou galões, na coluna Unidades da planilha Variáveis, pois as unidades não são utilizadas antes de definidas. Movimentamos o cursor para a planilha Regras pressionando [;] e depois digitando =U (de "unidade"). O TK!Solver substitui a planilha Regras, na janela inferior, pela Unidades. Esta tem quatro colunas: From (de); To (para); Multiply by (multiplicar por); e Add Offset (somar a constante).

Com o cursor sob FROM, podem-se introduzir as unidades que o TK!Solver deve reconhecer e os fatores de conversão, conforme a *Tela 2*.

Tela 2: Conversão de unidades

Digita-se =R para que volte à tela a planilha Regras, e a seguir [;], que move o cursor para a planilha Variáveis. Então, introduzem-se na coluna Unidades os nomes que definimos: m (milhas) para Distância; h para Tempo; m/h para Velocidade; g (galões) para Combustível; e m/g para Milhagem. Podemos agora apagar todos os valores atuais, substituindo-os por outros como: 1.247 para Distância, 22,5 para Tempo e 43,9 para Milhagem. Pressionamos [!] e a tela mostrará os valores no sistema métrico. *(Tela 3.)*

Tela 3: Conversão ao sistema métrico

Colocamos o cursor sobre o m na coluna de unidades, digitando km (quilômetros). Ao pressionarmos [Return], o TK!Solver converterá o valor 1.247 da Distância de milhas para quilômetros, passando esse valor para 2.006,423.

TABELA DE VALORES	
ENTRADA	NOME
500	Distância
8.5	Tempo
	Velocidade
14	Combustível
	Milhagem

OPERADOR DISCRETO

O gerente
Todo computador tem alguma forma de sistema operacional um programa que gerencia o funcionamento da máquina e controla todos os dispositivos ligados ao sistema

O sistema operacional trabalha "nos bastidores" de um computador: muitos usuários nem sabem que ele está presente na máquina. No entanto, é uma parte vital, constituindo o elo entre o hardware e o software.

As linguagens de alto nível permitem que o operador se isole das operações internas da CPU e contribuem para o maior intercâmbio de programas de uma família a outra. Desde que uma linguagem seja razoavelmente padronizada, as instruções devem valer para qualquer máquina que a aceite. O interpretador ou o compilador que processa o código-fonte de linguagem de alto nível também é um programa, devendo ser introduzido na memória principal antes de poder converter o código-fonte em instruções de código-objeto prontas para execução. O BASIC em ROM está presente na memória em caráter permanente, apto para utilização logo que se liga a máquina.

No entanto, para operar um computador, é preciso mais que um programa capacitando-o a converter o código-fonte em código de máquina. Deve haver outro programa que funcione em segundo plano, "nos bastidores", e se ocupe da organização interna. Consideremos, por exemplo, o problema de fazer aparecer no vídeo uma letra digitada no teclado. É preciso que haja, em algum ponto da memória, um programa que mande a CPU verificar constantemente o teclado para checar se qualquer tecla foi acionada. Se isso acontecer, o programa tem de identificar a tecla e, então, instruir os circuitos de vídeo para produzirem o padrão correto de pontos, na seqüência certa, para saída em vídeo. Tais atividades são chamadas "transparentes", isto é, invisíveis para o usuário.

Por exemplo, quando um comando é chamado para gravar um arquivo em fita cassete, o programador não se preocupa em saber como os dados se convertem numa forma adequada para armazenamento. Isso se realiza pelo sistema operacional — um programa gerenciador em processamento contínuo, encarregado de supervisionar todas as atividades do computador.

Surge uma leve fonte de confusão quando se utiliza um sistema de computação pequeno, baseado em ROM com BASIC residente, pois, nesse caso, o BASIC e o sistema operacional muitas vezes estão na mesma ROM. Em sua forma mais simples, uma ROM interna contém todo o soft-

ware básico necessário para operar o sistema, independentemente de programas aplicativos (jogos, processadores de texto etc.) carregados ou escritos pelo usuário.

Parte da ROM contém o interpretador, isto é, o código necessário para converter programas aplicativos escritos em BASIC para a linguagem de máquina (compreensível pelo microprocessador); outra parte contém o código necessário para introduzir e modificar programas escritos pelo usuário (o editor); e uma terceira parte contém o monitor, o software de organização interna necessário para varrer constantemente o teclado, apresentar na tela caracteres e desenhos, aceitar dados de fita cassete, alocá-los nos pontos corretos da memória e assim por diante.

O termo "monitor" não se refere a qualquer componente de um sistema de vídeo: é quase um sinônimo de "sistema operacional". Em sua versão mais simples, o monitor pouco mais pode fazer do que aceitar instruções em linguagem de máquina, colocá-las no ponto correto da memória e supervisionar sua execução.

No outro extremo estão os sofisticados computadores profissionais baseados em discos, que dispõem de poderosos sistemas operacionais.

Um sistema baseado exclusivamente em discos flexíveis em geral tem poucas instruções armazenadas na ROM em caráter permanente, à exceção de um autocarregador e de algumas rotinas de organização interna. Quando o computador é ligado, o autocarregador contém instruções em linguagem de máquina suficientes apenas para instruir a CPU no sentido de acessar uma unidade de discos e carregar na memória principal o sistema operacional. Este, carregado na RAM, constitui o DOS (Disk Operating System, "sistema operacional de disco"), de recursos bem maiores que os sistemas baseados na ROM. O DOS amplia as funções normais de organização interna, acrescentando comandos que atuam diretamente sobre os arquivos armazenados em discos. Os comandos de um DOS podem, por exemplo, listar os nomes dos arquivos contidos no disco, eliminá-los ou renomeá-los e copiar arquivos em disco na memória principal ou em outros discos.

Mais convencionais, os sistemas de operação baseados em ROM quase nunca dispõem desses comandos de processamento de arquivo. Em certos casos, podem ter simplesmente um comando para carregar arquivos nomeados a partir de fitas ou armazenar em fita um arquivo sob determinado nome. Sistemas operacionais sofisticados conhecem o endereço exato em que se armazena qualquer arquivo, seja o meio um disco ou uma fita. Sistemas menos avançados podem ser capazes apenas de fazer uma pesquisa sistemática em todos os arquivos existentes, até encontrar o nome desejado e então carregá-lo; ou gravar um arquivo sob determinado nome na fita cassete, no ponto em que esta estiver quando o comando for chamado.

Mas não existem apenas os sistemas convencionais, baseados na ROM, e os baseados em discos flexíveis; há uma vasta gama de computadores intermediários. Um bom exemplo é o Apple II, apto a manipular arquivos em fitas ou discos, muitas vezes a partir da utilização dos mesmos comandos. O sistema operacional

Controles operacionais

No sistema operacional CP/M, o programa do usuário é processado por meio do BDOS, coletânea de rotinas que proporciona recursos e funções padronizados para o programa. Sendo o BDOS o mesmo em qualquer máquina, um programa que o utilize funciona em qualquer sistema CP/M. Quando o BDOS precisa efetuar alguma tarefa, chama o BIOS, sistema adequado para cada máquina e que representa o elo de software entre o BDOS e os componentes físicos do computador. O CP/M também inclui o CCP (Command Control Program, "programa controlador de comando"), que permite a introdução de comandos para rodar programas e, de modo geral, controlar o sistema.

é residente em ROM, mas está fisicamente separado; pode-se imaginá-lo como um sistema operacional que foi carregado na memória a partir de dispositivos externos de armazenamento. Suas funções incluem comandos de gerenciamento de arquivos bem mais avançados que os encontrados no Spectrum e em outros computadores baseados somente em ROM.

Compatibilidade

A capacidade de usar o mesmo software em mais de uma família de computadores é conhecida como "compatibilidade". Nela, em geral, existem dois aspectos. O primeiro consiste em diferentes processadores requererem diferentes conjuntos de instruções para efetuar operações equivalentes. Assim, as instruções em linguagem de máquina necessárias para, por exemplo, somar o conteúdo de dois endereços de memória teriam uma forma se fossem escritas para o processador 6502, usado no Apple, e outra inteiramente diversa para o Z80, o chip do CP 500. O problema da conversão de linguagens de alto nível em linguagem de máquina é, contudo, de responsabilidade do interpretador ou compilador utilizado: CPUs diversas requerem interpretadores e compiladores diferentes.

O segundo aspecto a afetar a compatibilidade do software está no fato de que o uso da mesma CPU — no Unitron AP II e no Microdigital TK 2000, por exemplo — não impede a manifestação de outras dificuldades: uso de diferentes endereços para a memória de vídeo; necessidade de diferentes códigos para movimentar o cursor; existência de diferentes recursos de entrada e saída etc.

Para fazer frente ao problema foram desenvolvidos DOSs genéricos, aptos a permitir que o software escrito para, digamos, um Z80 baseado em disco seja processado em qualquer outro computador Z80 baseado em disco que tenha o mesmo sistema operacional. O mais conhecido DOS é o CP/M (Control Program/Microprocessors, "programa de controle para microprocessadores").

Os DOSs do tipo CP/M são, em essência, um desenvolvimento de sistemas operacionais e monitores mais específicos de certas máquinas, mas representam um avanço importante em termos de compatibilidade de software. Um programa escrito para processamento com CP/M, com MS-DOS ou outro sistema operacional genérico, será processado por qualquer computador com esses sistemas, desde que o software não tente fazer uso de nenhum recurso especial (efeitos sonoros, por exemplo) próprio de determinada máquina. O próprio software do sistema operacional é fornecido em forma padrão por seus criadores ao fabricante de computadores. O fabricante de hardware apenas reescreve um pequeno trecho do programa que depende da máquina a ser utilizada.

Os DOSs variam consideravelmente em termos de complexidade e capacidade; o CP/M e o MS-DOS estão entre os mais simples. Compõem-se, no fundamental, do processador de comandos, do BDOS (Basic Disk Operating System, "DOS básico") e do BIOS (Basic Input/Output System, "sistema básico de entrada e saída"). Dessas partes, a única que importa para a compatibilidade é o BIOS: um trecho separado do programa, com todas as rotinas necessárias ao gerenciamento de periféricos, incluindo o terminal de vídeo e o teclado, e que deve ser configurado especialmente para cada novo modelo de computador. Qualquer programa processado sob o controle desse sistema operacional interfaceia com o computador por meio do BIOS. Este vai comandar a leitura de caracteres a partir do teclado, a apresentação de caracteres no vídeo ou na impressora, o endereçamento de discos e inúmeras outras tarefas.

A chave do sucesso
O Osborne-1 deve boa parte de seu êxito de vendas ao fato de ter sido o primeiro microcomputador portátil a usar o sistema operacional CP/M. Cada máquina vem acompanhada por alguns dos mais populares programas para CP/M, a exemplo do WordStar, do SuperCalc e do MBASIC.

O BDOS consiste nas partes do sistema operacional não específicas ao nível de dispositivo — rotinas genéricas de gerenciamento de vídeo, impressoras, unidades de disco etc. — e que não precisam ser alteradas para novas implementações. Juntos, BDOS e BIOS correspondem aproximadamente ao monitor ou ao sistema operacional encontrados em computadores baseados na ROM.

O processador de comandos é a parte do programa que gerencia os comandos do sistema operacional digitados no teclado. Entre eles estão os comandos que carregam arquivos do disco para a memória principal, listam os nomes dos arquivos presentes no disco e eliminam ou renomeiam arquivos em disco.

HISTÓRIA DO MICROPROCESSADOR

Dos milhões de microprocessadores usados em todo o mundo, a maioria é de modelos 8088, 6502 e Z80. Embora curta, a história dos microcomputadores mostra como se estabeleceu essa hegemonia.

Os computadores operados por um único microprocessador surgiram quase por acidente. Em 1972, a Intel, fabricante americana de chips, foi encarregada pela Datapoint de desenvolver um chip que substituísse o processador discreto (específico), formado por grande número de chips TTL (Transistor-Transistor Logic, "lógica transistor a transistor") e utilizado nos terminais de computadores.

O produto desenvolvido foi o 8008, um chip capaz de processar dados a 8 bits por vez, mas que apresentava sério inconveniente: trabalhava muito devagar. A Datapoint decidiu não lançá-lo, mas o potencial do 8008 como uma CPU para computadores de uso geral não passou despercebido. Surgia o computador portátil, de baixo custo.

As evidentes limitações do 8008 levaram a Intel a projetar um modelo mais aperfeiçoado. Um novo chip, o 8080, logo se tornou a força dominante no mercado.

O lançamento do 8080 praticamente coincidiu com o do 6800, microprocessador também de 8 bits desenvolvido pela Motorola, concorrente da Intel. As filosofias de estrutura dos dois produtos divergiam bastante, mas ambos eram potentes e capazes de servir de base a um projeto de microprocessador.

O desenvolvimento seguinte veio em 1974, quando Gary Kildall, então presidente da Digital Research, projetou para a Intel o CP/M, um sistema operacional em discos. Isso permitiu que os computadores baseados no 8080 fossem usados com as unidades para discos flexíveis, recém-introduzidas no mercado. Apesar disso, a Intel rejeitou o sistema operacional CP/M, alegando que o software existente era mais que suficiente para os computadores de grande porte da época.

No entanto, os computadores menores estavam se tornando cada vez mais populares, e neles o CP/M facilitava muito a manipulação de arquivos. Isso assegurou por muito tempo o domínio do mercado pelo 8080 e colocou o Motorola 6800 em relativa obscuridade. Houve diversas tentativas de produzir sistemas operacionais em disco equivalentes para o 6800, mas a atenção já estava toda voltada para o 8080.

A escolha vital
A maioria dos micros pessoais utiliza um microprocessador 6502 (por exemplo, o Micro Engenho, da linha Apple) ou um Z80 (caso do TK85, da linha Sinclair, do TK90X, compatível com o Spectrum, e de outros micros da série TK).

À medida que crescia o mercado de produtos baseados em microprocessadores, aumentou bastante a competição entre os fabricantes de chips para o desenvolvimento de novos modelos. Deparavam, porém, com a relutância dos compradores em aceitar algo novo, a menos que fossem oferecidas nítidas vantagens. Os investimentos em hardware e software também contribuíam para desestimular a adoção de processadores incompatíveis com a tecnologia existente.

Um lance de genialidade abriu espaço para um novo modelo de chip — o Z80. A Zilog nasceu de uma equipe de engenheiros que trabalhara no 8080 da Intel; eles perceberam que o conjunto de instruções binárias do 8080 poderia ser ampliado. Não haviam sido exploradas todas as combinações possíveis de 1 e 0 que esse chip poderia reconhecer como instruções. Incorporando combinações binárias "esquecidas" pela Intel, a Zilog criou um microprocessador que operava de modo idêntico ao 8080 quando recebia instruções típicas desse microprocessador, mas que tinha melhor desempenho. Dessa maneira, tornou-se possível usar o software desenvol-

A evolução dos chips

Este diagrama mostra como os microprocessadores se desenvolveram e também algumas das máquinas em que foram utilizados. Diversos chips pouco conhecidos aparecem nos micros menos populares. O Apple III talvez seja o único micro profissional que use o processador 6502. O Olivetti M20 é o único micro de uso geral equipado com o chip Z8000. Em ambos os casos, a escolha de um microprocessador incomum e a conseqüente falta de software inibiram o êxito da máquina. Paralelamente o IBM PC e outros computadores de enorme sucesso estendiam sua popularidade aos chips utilizados por eles.

Intel

4004 e 8008 Projetado para substituir processadores discretos compostos por grande número de circuitos integrados TTL (Transistor Transistor Logic, "lógica transistor a transistor"), o chip 4004 podia manipular apenas palavras de 4 bits. O 8008, desenvolvido logo em seguida, processava os dados de 8 em 8 bits. Reconhecendo seu potencial, amadores e engenheiros começaram a utilizar esse chip na construção "caseira" de microcomputadores.

Motorola

6800 Este microprocessador tinha recursos semelhantes aos do 8080, desenvolvido pela Intel mais ou menos na mesma época, mas seu projeto e modo de operação eram totalmente diversos.

MOS Technology

Zilog

6809 Aperfeiçoando seu 6800, a Motorola criou o 6809, talvez o mais versátil dos chips de 8 bits. Mas os concorrentes já dominavam o mercado, e esse novo chip foi utilizado em poucas máquinas.

Dragon 32

PET

6502 A tecnologia MOS criou seu próprio chip de 8 bits, que, embora incompatível com o 6800, em grande parte derivava dele. Seu baixo custo atraiu o interesse de amadores e projetistas. O resultado foi seu emprego na primeira geração do Apple e do Commodore PET, máquinas de enorme sucesso comercial. O chip permanece popular, utilizado no Unitron, no Micro Engenho, no TK 2000 e em outros computadores.

8080 Com a criação do sistema CP/M, tornou-se o primeiro chip apto a ser empregado em micros tanto de uso pessoal quanto profissional. Mais tarde, a Intel criou o 8085, com funções adicionais e menos chips auxiliares. O modelo 8085 é encontrado numa versão CMOS de baixo consumo, presente em portáteis.

TRS-80 Model 100

Z80 Uma equipe de projetistas deixou a Intel para formar a Zilog, que produziu esta versão aperfeiçoada e plenamente compatível com o 8080.

TRS-80 Model 1

Apple III

68000 A eficácia do chip de 16 bits da Motorola foi amplamente reconhecida. A Sinclair utilizou em seu modelo QL o chip 68008, versão reduzida do 68000. Mas seu êxito acabou limitado pela falta de software barato e pelo domínio do 8088.

QL

Lisa

IBM PC

8088 e 8086 Uma versão reduzida, o 8088 pode usar chips auxiliares mais antigos; seu emprego no IBM PC fez dele o mais popular chip de 16 bits. O 8086, de melhor desempenho e totalmente compatível com o 8088, é hoje utilizado na maioria das máquinas.

Z8000 O primeiro chip de 16 bits da Zilog não se saiu muito bem no mercado dos micros. A segunda tentativa da empresa foi o Z800, que promete total compatibilidade com o Z80, portanto também em relação ao 8080, da Intel.

Olivetti M20

vido para o 8080 e contornar o problema da compatibilidade.

Além dessas inovações, introduziu-se importante vantagem comercial. O chip da Intel dependia de um chip gerador de sincronismo (clock) e de um chip de controle de sistema, enquanto o Z80 combinava num único chip toda a lógica necessária a um computador. Embora fosse relativamente caro, a capacidade de substituir vários outros chips tornava-o vantajoso para muitos fabricantes.

O chip 6800 não teve a mesma sorte em termos comerciais. A Motorola criou, afinal, um sofisticado microprocessador de 8 bits, o 6809, que representava um aperfeiçoamento do 6800. Quando o 6809 chegou ao mercado, porém, uma companhia rival, a MOS Technology, havia lançado o 6502, chip que constituía um aperfeiçoamento ainda maior do 6800. O modelo 6502 é o mais popular de uma série de microprocessadores conhecida como 6500, cujos integrantes usam o mesmo conjunto de instruções, mas diferem em potência e recursos.

O Z80 incorpora o conjunto de instruções do 8080 e pode substituí-lo num sistema de computação, embora exija, para tanto, modificações estruturais. O projeto do 6502, por sua vez, segue uma filosofia muito próxima à do Motorola 6800. Qualquer programador do 6800 estaria "em casa" com o conjunto de instruções do 6502. Todavia, este não é compatível com o 6800 quanto aos requisitos de hardware, software ou possibilidade de substituição chip por chip. Por tudo isso, era pouco provável que viesse a alcançar êxito imediato, exceto por outro golpe de sorte: foi adotado no computador Apple, muito bem-sucedido.

Quando a Apple surgiu, os micros portáteis eram dominados pelos projetos de bus baseados no S-100. (O bus diz respeito ao número de bits de um computador. Esse número pode ser dado pelo bus interno — por exemplo, 32 bits no 6800 — ou pelo bus externo — 16 bits, no mesmo 6800. O valor dos bus varia de acordo com o microprocessador; nesta matéria, não havendo indicação em contrário, adota-se o valor do bus interno.)

O S-100 é uma padronização da conexão entre elementos, na qual sinais eletrônicos se transmitem pela ligação de placas. Os micros baseados nesse sistema dispõem de inúmeras placas — a principal, que transporta a energia e os sinais, e várias outras, uma para cada função. Um sistema S-100 reduzido ao mínimo exigiria uma fonte de energia, a placa principal, outras para a CPU, a memória, o vídeo e, provavelmente, mais placas para a impressora e a unidade de disco. Nesse sistema, cada placa necessita de um conector, e o conjunto de conectores requer mais uma placa para a conexão geral. Tudo isso o torna bem mais caro que o sistema Apple, que contém, numa única placa, a CPU, o vídeo, a memória e as saídas para as demais expansões — conhecidas por slots.

A grande conquista dos criadores do Apple foi a planilha eletrônica VisiCalc. Esse aplicativo conquistou enorme popularidade entre os empresários: era mais fácil e mais rápido processar previsões financeiras com ele do que com calculadora, lápis e papel. O VisiCalc acarretou vendas em massa do computador Apple. Em conseqüência, o 6502 tornou-se um dos microprocessadores mais importantes.

Redução de chips
Chips mais sofisticados reduzem o número de componentes numa placa de circuitos. Quando a Apple aperfeiçoou o Apple II, a nova versão IIe tinha metade dos chips mais importantes.

Enquanto o chip 6502 consolidava seu domínio na área dos computadores de 8 bits, começavam a surgir no mercado os computadores de 16 e 32 bits. A Intel oferecia-lhes o 8086 e o 8068 (microprocessadores com bus interno de 16 bits e externo de 8 bits). A Motorola produzia o 68000 e a Zilog, o Z8000 (ambos com bus interno de 32 bits e externo de 16 bits). Nenhum dos modelos é compatível com seus predecessores de 8 bits. A Digital Research e a Microsoft criaram sistemas operacionais para os 8066/8088 (CP/M e MS-DOS, respectivamente) da Intel, mas a Zilog e a Motorola ainda não dispõem de sistemas operacionais adequados. Além disso, a adoção pela IBM do 8088 em seu PC consolidou a preferência pelo chip da Intel.

A luta entre os chips de 16 bits pelo domínio do mercado promete reeditar a história dos chips de 8 bits. O 8086 e sua versão reduzida, o 8088, tornaram-se padrões, da mesma forma que o Z80 e o 6502. Os principais motivos são o software de apoio fornecido pelos sistemas operacionais MS-DOS e CP/M-86 e sua escolha pelos micros de alta vendagem, sobretudo o IBM PC. O chip Zilog Z8000 tem sido utilizado apenas num micro de emprego geral, o Olivetti M20. Depois de enfrentar sérias dificuldades para fornecer software a esse micro, a Olivetti lançou uma placa conectável contendo um microprocessador 8086, possibilitando seu funcionamento com software para o MS-DOS e o CP/M-86.

Apesar do desenvolvimento acelerado das máquinas que utilizam microprocessadores de 16 bits, mais velozes e potentes, a maioria dos computadores se baseia nos chips de 8 bits, seja o Z80 ou o 6502.

POLIMAX

Equipamentos dedicados
O sistema Poliscriba 301 WP é o principal produto da Polymax na área do processamento de textos — uma especialidade da empresa, desde sua fundação.

Teclado inteligente
Além de operar como terminal para sistemas de grande porte, o Poly 201 DE — criado especialmente para aplicações financeiras — também pode ser usado como um micro convencional autônomo.

A Polymax teve sua implantação aprovada, em 1977, pela extinta Capre (Coordenadoria de Atividades de Processamento Eletrônico), precursora da SEI (Secretaria Especial de Informática).

A Polymax Sistemas Periféricos foi fundada, em agosto de 1977, pelos engenheiros eletrônicos Lawrence Huang e Marcos Lee. Pouco depois, entrou para a sociedade Sheun Ming Ling, que financiou e dirigiu a empresa até julho de 1984. Nesse ano, a companhia Invesplan assumiu o controle acionário da Polymax, efetuando alterações na estrutura da empresa.

Para aumentar o capital, foram colocadas no mercado novas ações da empresa, agora com a razão social modificada para Polymax Informática. As mudanças na composição do capital e da diretoria não afetaram a continuação das atividades da firma.

Até o início de 1979 a principal preocupação da empresa esteve voltada para sua capacitação tecnológica e formação de uma equipe técnica. Nessa fase desenvolveram-se dois protótipos, dos quais resultaram o Poly 101 HS, micro dedicado ao processamento de textos, e o Poly 101 SS, para processamento de dados. Ambos começaram a ser comercializados em janeiro de 1979.

Dois anos depois foram substituídos por duas novas versões: Poly 201 WP e Poly 201 DP. A produção do primeiro cedeu lugar à do Poly 301 WP. Muito semelhante ao anterior, o projeto desse novo equipamento destinava-se a elevar o índice de produtividade na edição, no tratamento e no gerenciamento de textos.

Por suas características, o Poly 301 WP adapta-se aos sistemas de rede local e à automação de escritório. Sua capacidade de armazenamento também é superior à dos modelos anteriores, permitindo o uso de discos flexíveis de 8 polegadas ou discos rígidos tipo Winchester.

Durante o ano de 1982, a Polymax consolidou sua linha de equipamentos voltada ao processamento de textos e deu início a projetos de software para a área de teleprocessamento. Nessa linha se enquadra o lançamento do Sistelp (Sistema de Teleprocessamento Polymax), para operar com protocolos assíncronos e síncronos.

No mesmo ano, o mercado passou a contar com mais um microcomputador, o Maxxi, compatível com a linha Apple II Plus. O sistema multiusuário Poly 910 NET, lançado em abril de 1983, conectava até oito estações, configuradas por micros ou terminais inteligentes, permitindo o acesso a arquivos e troca de mensagens entre usuários (“correio eletrônico”).

Nessa época, a Polymax também desenvolveu dois sistemas para atender o setor financeiro: o Poly 201 DE (Data Entry) e o Poly 105 MT. O primeiro é produto híbrido, empregado como terminal para entrada de dados do sistema central e como microcomputador. O segundo consiste num terminal inteligente.

Na linha dos periféricos, o produto mais importante é a impressora Polyprint 90, com velocidade de 90 cps. Ainda em 1983, chegava ao mercado o PEC 76, simulador de unidades de controle IBM 3276.

Já sob a coordenação da Invesplan, em abril de 1985, houve o lançamento do sistema Poly 920 NET, uma rede do tipo barramento (bus) que admite interconexão de até 255 estações configuradas por micros da Polymax ou terminais inteligentes.

NOVA MÚSICA ELETRÔNICA

No começo, a música eletrônica produzia apenas sons muito simples, com osciladores discretos. Hoje, a tecnologia digital permite complexas matrizes sonoras: de pingos de chuva à imponência de uma orquestra.

A imaginação não tem limites. E, entre os que reafirmam essa assertiva todos os dias, os músicos aparecem com destaque. Sua capacidade de combinar sons e emocionar é tão poderosa que consegue nos levar ao riso ou às lágrimas. E para fazer isso eles vêm usando, ao longo do tempo, todo tipo de instrumento, dos tradicionais aos eletrônicos.

Entre os últimos, o oscilador é considerado fundamental. Sua construção baseia-se no fato de que, quando um objeto vibra, o ar em torno dele se expande e se contrai com muita rapidez, criando uma onda que o ouvido e o cérebro interpretam como som. Da mesma forma, quando aplicamos uma corrente elétrica a um fio condutor por meio de um modulador (uma bobina, por exemplo), o metal vibra, produzindo a mais simples forma de onda: a senóide.

Essa pequena unidade geradora de som é conhecida como um oscilador. O controle por variação de tensao foi, desse modo, durante décadas, o principal método usado na produção de música sintetizada.

Há vários anos, os estúdios de gravação têm inúmeros e diferentes equipamentos de processamento sonoro, e não raro um conjunto de unidades de filtragem e reverberação é tido como elemento essencial de um estúdio. Do mesmo modo, os tecladistas dos anos 70 costumavam apresentar-se completamente cercados por sintetizadores, cada qual com uma infinidade de controles. Ao considerarmos o que acontece num estúdio de gravação bem equipado, seria útil lembrar um jogo chamado cadeia-de-mensageiros. Nesse jogo, uma frase é passada de pessoa a pessoa e a última informa o que ouviu. Verifica-se então que uma frase clara e direta no início acabou transformada numa porção de palavras absurdas. Num estúdio de gravação, ocorre algo semelhante: a frase original seria um conjunto de sons e, ao invés da corrente de pessoas que ouvem e reproduzem versões deturpadas da original, tem-se um grupo de unidades de processamento sonoro, cada uma controlável e capaz de cumprir tarefas específicas.

Qualquer pessoa que tenha de trabalhar com esse equipamento provavelmente usará uma mesa de controle para fazer as conexões entre as unidades ou as ligará diretamente. Os controles de cada uma são geralmente calibrados à mão por um engenheiro de som — e podemos imaginar os problemas de sincronização e comunicação que isso tudo implica.

Tecnologia artistica
Foi para gente como Egberto Gismonti, que não para de fazer experiências com sons, que se empregou a tecnologia digital em instrumentos.

Controle central
Esta interface conecta microcomputadores a sintetizadores digitais: é a unidade processadora MIDI MP401, da Roland. Ela controla todas as funções externas de sincronização, entrada e saída de gravadores, além da saída de sinais para o sintetizador e o computador rítmico. Significa que o computador ao qual se liga responde apenas pelo envio e recepção de instruções e pelo gerenciamento do espaço na memória. Esta interface foi produzida inicialmente para os micros Apple II e II*e*, IBM PC e para o Commodore 64.

Na década de 70, os tecladistas enfrentavam outros problemas. A dificuldade imediata não era produzir uma seqüência de sons de cada instrumento — para isso, bastava o músico mover a mão de um teclado para outro. Também não havia problemas para tocar dois teclados ao mesmo tempo. E, para ajudar, a maior parte do trabalho em estúdio não era feita de uma vez: os sintetizadores eram gravados separadamente e cada parte sobreposta à anterior, por meio dos diferentes canais de um gravador. Contudo, a evolução mais significativa ocorreu nas unidades de geração de som dos sintetizadores e nas técnicas instrumentais em geral. Bom exemplo disso foi o aparecimento do sintetizador de baixos controlado pelo teclado.

Nessa época, os baixistas de funk atingiram um nível tal que rivalizavam com os guitarristas. No fim da década, no auge desse estilo, surgiram os baixos sintetizados em teclados. Para o tecladista, isso trouxe novos problemas. Ele teria de assumir também as funções de baixista, trabalhando com o baterista para "amarrar" o ritmo e deixar de fazer música "de teclado". E, à medida que novos sintetizadores analógicos foram surgindo no mercado, outros tipos de sons — trompete, saxofone e bateria — tornaram-se disponíveis. Cada vez mais os tecladistas adotavam um dispositivo simples para lidar com novas responsabilidades: o seqüenciador.

Trata-se de um dispositivo que coordena linhas melódicas (ou seqüenciais), utilizando chaves analógicas (potenciômetros) ativadas por um controlador de tempo (clock). Este, por sua vez, comanda o oscilador na produção de tons de freqüências diferentes. Os seqüenciadores determinam também a duração — curta ou longa — dos sons, por meio do controle de portas ou do tempo de duração do chaveamento. Com isso, podem-se criar padrões de ritmos e compor estruturas melódicas. O objetivo do seqüenciamento é garantir que o intervalo ocorra no mesmo instante, em cada repetição da seqüência.

No final dos anos 70, poucos sintetizadores possuíam recursos completos de seqüenciamento, mas os músicos logo começaram a usar tudo o que havia de disponível. A música de discoteca produzida por Giorgio Moroder e Donna Summer é, em grande parte, de seqüenciador — especificamente do Micro Composer MC-4, da Roland. E artistas brasileiros de vanguarda, co-

Criando um ritmo

Pode-se produzir som eletrônico de duas maneiras: programando osciladores, filtros, geradores de envoltório e de ruídos — muitas vezes usam-se ruídos "brancos" para gerar sons em micros — e também captando sons reais por meio de um dispositivo de conversão de analógico para digital e vice-versa. Ou seja, faz-se uma amostragem dos sons, que serve para criar uma espécie de gabarito. A partir deste, pode-se montar um modelo de onda sonora dificilmente conseguida por outros meios. Tente sintetizar o ruído de uma caixa de bateria em seu micro para ver a dificuldade

Analógico
A saída de um microfone é apenas uma corrente com tensão variável, análoga ao som que a produziu.

Digital
Um conversor digitaliza a onda numa série de valores de voltagem. Quanto maior o número de valores obtidos, mais preciso será o modelo digital.

mo Egberto Gismonti, usando sintetizadores, desenvolveram um estilo completamente novo para se ajustar aos novos equipamentos. Os novos seqüenciadores digitais podiam controlar seqüências inteiras, para serem iniciadas ou interrompidas pela mudança de um controle. Durante a seqüência, o músico podia se afastar do teclado, dançar ao ritmo da música e então voltar a outro teclado.

Quando surgiram os sintetizadores digitais, os músicos descobriram que suas habilidades para usar seqüenciadores poderiam ser desenvolvidas ainda mais. Foi essa área do controle digital que despertou mais interesse em todos os músicos. Isso ficou demonstrado pelo sucesso do computador rítmico Linn, uma das primeiras unidades a utilizar o som amostrado — no caso, do baterista americano Steve Gadd. No Linn, gravaram-se os padrões de percussão em forma digital, do mesmo modo que se gravam os dados computacionais em discos. As seqüências resultantes de 1 e 0 são então codificadas em chips de ROM. A grande vantagem em se digitalizar o som está no resultado, que pode ser alterado em termos de tempo, ritmo, volume etc.

O estúdio de gravação e o tecladista de palco que até agora usamos como exemplo têm várias características em comum. Além de gravar música, o trabalho de estúdio reflete o crescimento na área do vídeo ocorrido nos últimos anos, trazendo um novo estilo e uma nova consciência do trabalho com imagens. Os produtores exigem que a música de cada videoclip contenha esse aspecto.

Se a música de um videoclip for produzida com instrumentos tradicionais, a sincronização com a imagem será feita pelo mesmo processo usado em cinema. Contudo, se vier de sons individuais e seqüências produzidas por sintetizadores digitais, muita coisa poderá dar errado. Vamos imaginar que, num certo videoclip, um vaso caia no chão e se quebre. O músico que está trabalhando na trilha produziu uma seqüência que se acelera até o ponto em que o vaso cai, e programou um acorde percussivo para o momento em que chega ao solo. A gravação começa, mas logo fica claro que a aceleração foi mal calculada, e a seqüência termina enquanto o vaso ainda está na mesa. O músico tenta então o acorde de percussão, que está gravado em outro canal. A gravação recomeça e, dessa vez, o acorde é gravado com 1 microssegundo de atraso. Os músicos necessitam, portanto, de uma forma de interligar os instrumentos digitais de modo que tudo aconteça na hora certa.

O tecladista tem problemas similares nas apresentações ao vivo. Seu equipamento pode incluir, digamos, dois sintetizadores digitais de marcas diferentes, além do Linn. Ele tem seqüências gravadas em todos, mas, como elas não estão no mesmo compasso, geralmente acaba colocando o Linn no automático e tocando a outra seqüência com as mãos. O resultado é que seu segundo sintetizador permanece inativo. Esse músico precisa de um meio de interligar seus instrumentos, para que todo o material seqüenciado seja acionado no tempo certo. E ele também necessita que o seqüenciador de seu primeiro sintetizador toque os sons pré-programados do segundo. Além disso, qualquer equipamento que utilize deve se adaptar a outros sintetizadores, além dos seus.

No próximo artigo, examinaremos a solução para isso tudo: a interface MIDI. Lançada em 1983, a partir de um acordo estabelecido entre os grandes fabricantes de hardware, essa unidade foi projetada para permitir que um determinado sistema digital (como um computador) controle outro (um sintetizador, por exemplo).

Dupla dinâmica
Donna Summer gravou um dos primeiros discos com ritmo produzido por sintetizadores. A idéia e o trabalho foram do produtor Giorgio Moroder

COMPUTADOR RÍTMICO

Teclas programadas
Um ritmo padronizado pode ser composto com um computador rítmico (Linn LM1, Roland TR-707, Oberheim DX), por meio do teclado. Na execução, o dispositivo verifica quais as teclas programadas, e reproduz a batida no tempo certo — o qual se pode mudar a qualquer instante.

RITMO DA BATIDA

A CHAVE DOS DADOS

Sistemas gerenciadores de bancos de dados organizam a informação em registros contendo campos de dados de vários tipos. Mas, para acessar determinado registro, devemos considerar o conceito de "chave".

Instrumento incorporado a um SGBD (Sistema Gerenciador de Banco de Dados), a chave ajuda a localizar registros específicos. Vamos considerar um manual de manutenção de automóvel. Quando queremos regular o carburador, sabemos que a informação sobre como proceder está em algum lugar do manual. Como pode estar na página 36, esse número passa a ser considerado a chave para a informação. Mas talvez não saibamos que o capítulo sobre essa peça se encontra na página 36; por isso, recorremos ao índice. Este dará o número da página, permitindo-nos ir direto ao ponto, sem necessidade de folhear cada uma das páginas.

É isso que ocorre com um SGBD. Cada registro num arquivo de banco de dados possui um único número de identificação (a chave primária). Mas, para encontrar determinado registro, você tem de escolher entre examinar um por um, seqüencialmente, até obter o dado procurado, ou, se já conhece o número do registro, ir diretamente a ele. Você também dispõe da opção de usar um dos campos como chave (no caso, uma chave secundária). Se tivermos um banco de dados sobre manutenção de automóveis, deveremos utilizar o campo NOME-PECA como chave.

A maioria dos SGBDs permite designar campos específicos como chaves. Quando um campo (NOME-PECA, no caso) for designado como campo-chave, o SGBD manterá uma tabela interna de palavras relacionadas com o campo especificado, junto com o número do registro apropriado (chave primária).

Quando você quer o registro que trata de carburadores, o SGBD procura na tabela NOME-PECA até encontrar CARBURADOR, verifica o número de registro correspondente e o acessa. Eis como se deve implementar tal esquema num SGBD simples escrito em BASIC:

```
INPUT 'ENTRAR COM CAMPO-CHAVE';CHA$
INPUT 'ENTRAR COM PALAVRA A
  PROCURAR';PAL$
GOSUB 20000 : REM SUB-ROTINA DE PESQUISA
PRINT REGISTRO
```

Uma entre várias matrizes foi selecionada, pelo uso de CHA$, e pesquisada por uma sub-rotina que usa o string PAL$ como chave de pesquisa. Se a sub-rotina de pesquisa for suficientemente poderosa, terá condições de localizar um registro mesmo que se cometam pequenos erros de digitação na chave. Uma rotina simples como essa não depende da existência de uma tabela de chaves. Técnicas de pesquisa direta nos registros serão suficientes.

Os primeiros SGBDs para microcomputadores eram simples programas que qualquer amador poderia escrever em BASIC. Hoje, os micros de 16 bits contam com poderosos bancos de dados, que até há pouco estavam além do alcance dos micros mais simples. Mas um grande impulso foi dado com o advento do Sinclair QL, com seu processador 68000. A Psion (fabricante inglesa de software), utilizando os poderosos minicomputadores VAX, desenvolveu, para o Sinclair QL, o banco de dados Archive, pondo ao alcance do usuário amador avançados recursos de manipulação de dados.

Como exemplo de SGBD, vamos criar um banco de dados trivial, com quatro registros, sobre manutenção de automóveis. Cada um deles consistirá em apenas dois campos, NOME-PECA e CONSERTO:

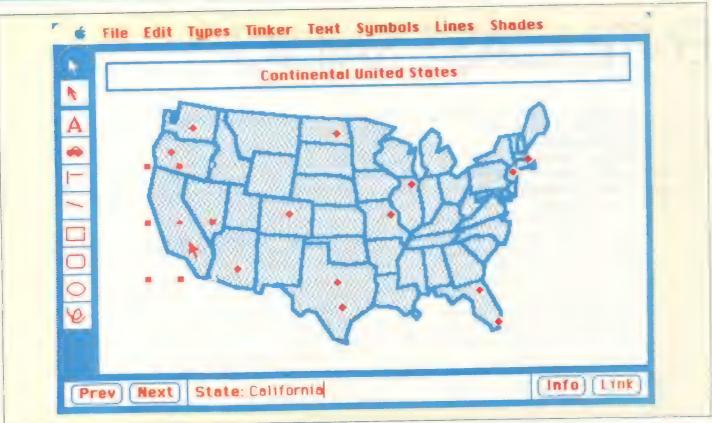
```
Carburador
Remover a tampa e girar o botão

Tanque de óleo
Abrir o tampão e encher com óleo

Bateria
Abrir tampas e encher com água destilada

Borrifador de água
Abrir a tampa e encher com água
```

Quando o usuário roda o Archive, a tela do QL divide-se em áreas: uma de comandos (na parte superior), outra de trabalho (no centro) e uma terceira para exibição (na parte inferior).

Para criar um novo arquivo de banco de dados, seleciona-se o comando CREATE. No caso do banco de dados para manutenção de automóveis, digita-se CREATE 'CARRO' e entra-se com os nomes dos campos:

```
NOME-PECA$
CONSERTO
```

O sinal $, com os nomes dos campos, indica serem estes alfanuméricos.

Para inserir registros no banco de dados, digite INSERT, a fim de que sejam exibidos os nomes dos campos na área de trabalho. Então você

pode entrar com os dados para cada campo. Pressionando a tecla [F5], um registro corretamente introduzido será acrescentado ao arquivo do banco de dados. Quando todos os registros que se quer introduzir no arquivo forem digitados, pode-se sair do modo INSERT acionando [Escape]. O encerramento do arquivo se faz pela digitação de CLOSE.

Para procurar um registro num arquivo, este deve inicialmente ser aberto. Pode-se fazer isso digitando o comando LOOK, que permite apenas examinar os registros, ou OPEN, que faculta pesquisas e alterações. Após ter aberto ou examinado um arquivo, você emite comandos simples, como FIRST (que exibe o primeiro registro no arquivo), LAST (o último), NEXT (o seguinte) ou BACK (o anterior).

Para pesquisar um registro, acrescenta-se um argumento a um comando — por exemplo, FIND'CARBURADOR'. Este buscará em todos os registros até localizar o que coincide com o string entre apóstrofos.

Operadores lógicos (E, OU e NAO) também podem ser usados, como no comando SEARCH NOME-PECA$ = 'CARBURADOR' AND CONSERTO = 'GIRAR'. Esse comando procurará no arquivo até encontrar um registro contendo o string CARBURADOR no campo NOME-PECA$ e o string GIRAR no campo CONSERTO$ do mesmo registro. Usa-se o operador OU a fim de que um registro seja localizado caso o string especificado para um campo ou o string especificado para o outro campo coincida com o nome procurado.

Em geral os registros entram num banco de dados fora de ordem, mas será necessário acessá-los e imprimi-los segundo alguma classificação. Suponhamos, por exemplo, que você se propôs introduzir num banco de dados todos os livros de uma biblioteca, por autor, título, editora e o padrão internacional para numeração de livros (ISBN, International Standard Book Number). A maneira mais simples de fazer isso é percorrer as prateleiras e introduzir os dados de cada livro encontrado. Mas, utilizando a capacidade de classificação inerente a qualquer SGBD, você pode ordenar os registros alfabeticamente (por autor, título etc.), ou por número (pelo ISBN).

É possível combinar uma classificação primária com outra, secundária. Suponhamos que você queira o arquivo impresso em ordem alfabética por autor. Cada registro será depois ordenado alfabeticamente pelos títulos das obras. Imagine uma lista muito maior, mas com estas características:

A herdeira, Sidney Sheldon
Memórias do cárcere, Graciliano Ramos
Infância, Graciliano Ramos
Ainda estamos vivos, J. M. Simmel
Se houver amanhã, Sidney Sheldon
Até o mais amargo fim, J. M. Simmel
São Bernardo, Graciliano Ramos

Ela pode ser ordenada por autor e, a seguir, pelos títulos mediante este comando:

```
ORDER AUTOR$:A,TITULO$:A
```

O A, nesse exemplo, significa "lista por ordem alfabética ascendente".

Os SGBDs transformam em trabalho leve tarefas que em geral exigiriam muito tempo, como localizar registros e reordená-los de maneira específica.

Os EUA na ponta dos dedos
O banco de dados Filevision destinava-se originalmente aos computadores Apple II e III. Mas teve de esperar até que o maior poder de processamento do Macintosh permitisse que suas notáveis e únicas características fossem plenamente apreciadas. O programa utiliza um mouse para acessar o banco de dados. Junto com o programa há um sofisticado pacote para desenho, que permite criar uma estrutura gráfica para o banco de dados. As formas e os símbolos que você desenhar são relacionados aos campos do banco de dados. A figura corresponde à tela principal de um banco de dados de informações sobre os Estados Unidos. Se apontarmos o mouse para a Califórnia, um clique simples dará uma breve lista de dados acerca desse Estado, um clique duplo abre um campo de texto que armazenará maior quantidade de informações. Os dados registrados podem ser relacionados e classificados da mesma maneira que em qualquer banco de dados. Por exemplo: o programa permite especificar um critério de pesquisa — digamos, todos os Estados com população superior a 10 milhões de habitantes — e então exibir os resultados traçando esses Estados, no mapa, com contornos mais espessos

ENTRADAS E SAÍDAS

O controle de entrada e saída é um dos aspectos importantes da linguagem ASSEMBLY. Vamos ver como funcionam os dois chips de interface mais usados com o processador 6809 — o PIA 6820 e o ACIA 6850.

O processador 6809, uma versão do 6502, não possui espaço específico para endereçamento ou instruções especiais de entrada/saída. Em vez disso, os chips de interface de I/O (Input/Output, "entrada/saída") utilizam o espaço de endereçamento normal do 6809 e são manipulados por meio de instruções de acesso à memória. Para o processador, esses dispositivos funcionam como se fossem parte da memória.

Apesar de sua simplicidade e rapidez, esse recurso ocupa um bloco de endereços, que, portanto, deixa de estar disponível para uso normal. Em conseqüência, embora o 6809 possua um bus de endereçamento de 16 bits capaz de endereçar 64 Kbytes de memória, fica restrito a cerca de 56 Kbytes, excluída a memória para controlar o hardware e o software. Esses chips de interface possuem uma complexidade tão grande quanto a do próprio processador e pertencem, em geral, à mesma família que ele, pois isso facilita sua conexão e controle.

Os dois chips mais utilizados com o 6809 são o PIA (Peripheral Interface Adaptor, "adaptador de interface periférica"), modelos 6802 ou 6821 — que controlam entradas e saídas paralelas —, e o ACIA (Asynchronous Communications Interface Adaptor, "adaptador para interface de comunicações assíncronas"), que opera com entradas e saídas seriais. Cada um deles possui numerosos registros e, para efeito de controle, basta tratá-los como se fossem posições normais da memória. Há três tipos de registro: de controle, de status e de dados.

Registros de controle. Exclusivos para gravação; armazenam valores que permitem programar o chip para as opções desejadas, tal como a de estabelecer a velocidade de transmissão de dados em bauds (1 baud = 1 bit por segundo).

Registro de status. São apenas para leitura, e seus valores indicam o status (estado) do chip. Esses valores mostrarão, por exemplo, se uma entrada foi recebida, se a última saída foi transmitida ou se houve algum erro.

Registros de dados. Contêm os dados que entram ou que saem e, portanto, são usados para leitura e gravação, simultâneas ou separadas.

Questões periféricas

As impressoras somente aceitam dados que [illegible]am no formato apropriado e na velocidade certa. Seria um desperdício a CPU se ocupar com tais questões relativamente triviais. Por isso ela envia os dados ao adaptador de interface periférica (PIA [illegible] apenas a co[illegible]

A fim de economizar espaço na memória, costuma-se gravar mais de um registro no mesmo endereço — por exemplo, um registro de status e um de controle. O que nele aparece depende de se estar lendo ou gravando. De modo análogo, um registro de dados de entrada e outro de dados de saída também podem compartilhar um mesmo endereço.

O PIA 6820 dispõe de seis registros e ocupa 4 bytes consecutivos no espaço da memória. O chip contém, de fato, duas portas (A e B) independentes, cada uma utilizando três registros. O lado do chip ligado aos periféricos possui oito linhas de dados e duas de controle para cada porta. Podem-se conectar as linhas de controle a outras, também de controle, no periférico, de tal modo que determinem o status. A linha 1 destina-se apenas a sinais de controle de entrada, mas a de número 2 é programável para receber ou enviar sinais de controle. Em cada porta temos:

- Registro de dados, que funciona tanto para entrada quanto para saída, uma vez que se pode acessar cada bit de modo independente.
- Registro de endereçamento, onde se usa cada bit para acessar o bit correspondente no registro de dados como entrada (0) ou saída (1).
- Registro combinado de controle/status, no qual os registros anteriores compartilham o mesmo endereço. O estado de um dos bits no registro de controle determina qual deles aparece nesse endereço. A tabela à esquerda dá o desvio do endereço acessado pelo chip para cada um dos registros.

	Registro	Desvio
Porta A	Dados A	[illegible]
	Direção de dados A	0
	Controle/Status A	1
Porta B	Dados B	2
	Direção de dados B	2
	Controle/Status B	3

Os bits no registro de controle/status são determinados pela tabela a seguir:

Bit	Função
7	Bit de status para a linha de controle 1; ativado com valor 1 ao receber um sinal de controle e automaticamente zerado quando se lê o registro de dados
6	Bit de status para linha de controle 2; como o bit 7
5	Determina o uso da linha de controle 2 para entrada (0) ou para saída (1)
4	Determina a natureza do sinal de controle na linha 2
3	Com a opção para entrada na linha de controle 2, o valor 1 nesse bit ativa a interrupção do bit 6; se a linha for ajustada para saída, esse bit ajudará a determinar a natureza do sinal
2	Seleciona entre registros de dados (1) e de direção de dados (0)
1	Determina a natureza do sinal de controle na linha 1
0	Um valor 1 nesse bit ativa a interrupção do bit 7

O primeiro dos nossos programas-exemplo inicializa e usa um chip 6820 para controlar uma impressora por meio da interface padrão Centronics. Essa interface especifica um grande número de linhas de controle, bem como as linhas de dados. Uma linha de controle, chamada "strobe", avisa a impressora que um caractere está a caminho. Essa linha conecta-se à linha de controle 2, que deve ser ajustada para saída. Outro sinal de controle da impressora, denominado "acknowledge", indica que ela está pronta para receber o caractere seguinte. Este deve ser conectado à linha de controle 1. As oito linhas de dados conectam-se às oito saídas de dados da porta PIA.

Para instalar a porta, devemos selecionar o registro de endereçamento dos dados e programar para saída todos os 8 bits. Depois, selecionamos o registro de dados e ajustamos a linha de controle 2 para saída. Para usar o chip, lemos de modo contínuo o registro de controle/status até um dígito 1 aparecer no bit 7, indicando que a impressora está pronta para um caractere. Podemos então gravar um caractere no registro de dados, que envia um sinal à linha de controle 2. O bit 6 assumirá o valor 1 quando o caractere for transmitido. Lemos, então, o registro de dados para limpar os bits 6 e 7 e repetir o processo até o último caractere. O processo de envio e recepção de sinais de controle entre o processador e os periféricos é conhecido por "handshaking".

Devemos supor que o endereço de base do PIA seja mostrado na tabela de endereços localizada na posição $3000. Na entrada da sub-rotina para a impressora, o registro A do processador contém o índice dessa tabela, e o registro Y, o endereço do string a ser impresso. Este é armazenado no formato normal, ou seja, com o byte indicador de tamanho em primeiro lugar. Há duas sub-rotinas, uma para preparar a porta e outra para imprimir o string.

ACIA 6850

O chip ACIA 6850 é um UART (Universal Asynchronous Receiver/Transmitter, "receptor/transmissor assíncrono universal"), utilizado para comunicação serial, que emprega o protocolo RS232 e, eventualmente, um modem. Possui quatro registros e ocupa dois endereços. Há cinco conexões com o chip, no lado dos periféricos: uma linha para os dados transmitidos, outra para os recebidos e três linhas para troca de informações de controle. Duas delas são para sinais de controle de entrada (DCD — Data Carrier Detect, "detecção de transporte de dados"; e CTS — Clear To Send, "limpeza para transmissão"), e outra é para sinais de saída (RTS — Request To Send, "pedido de transmissão"). Podem-se conectar essas linhas às de nomes semelhantes num periférico padrão RS232.

Registro	Desvio
Registro de controle	0
Registro de status	0
Transmissão de dados	1
Recepção de dados	2

Os registros ACIA são mostrados na tabela da margem esquerda. No de controle, o bit mais significativo (bit 7) serve para ativar as interrupções, possibilitando a recepção de dados. Os bits 5 e 6 ativam ou desativam interrupções na transmissão e determinam o tipo do sinal de controle enviado pela linha RTS. Os bits 2, 3 e 4 determinam o tamanho do "pacote" transmitido. No envio de um byte por uma porta serial, usam-se, em geral, pelo menos 10 bits. A transmissão começa com o bit inicial (start bit), que informa o envio de dados ao receptor. A unidade de dados transmitida possui 7 ou 8 bits e pode conter 1 bit de paridade. O bit extra de paridade auxilia a detecção dos erros de transmissão. Por fim, existem 1 ou 2 bits de parada (stop bit). As opções são:

Registro de controle			Número de bits de dados	Paridade	Número de bits de parada
Bit 4	Bit 3	Bit 2			
0	0	0	7	Par	2
0	0	1	7	Impar	2
0	1	0	7	Par	1
0	1	1	7	Impar	1
1	0	0	8	Nenhuma	2
1	0	1	8	Nenhuma	1
1	1	0	8	Par	1
1	1	1	8	Impar	1

Os dois bits menos significativos (0 e 1) determinam a velocidade de transmissão e recepção. Para isso, estabelece-se um divisor para o valor do clock. O 6850 não possui clock, devendo, portanto, ser complementado com um externo, ajustado em 1.760 Hz.

Registro de controle		Divisor do valor do clock
Bit 1	Bit 0	
0	0	1
0	1	16
1	0	64

A combinação que não é mostrada na tabela, quando ambos os bits são 1, provoca uma reinicialização geral no chip.

No registro de status, os bits possuem as seguintes funções:

Bit	Função
7	Interrupção requerida
6	Ativado com valor 1 ao ocorrer erro de paridade na recepção
5	Ativado com valor 1 quando há sobrecarga do receptor, isto é, se bits demais são recebidos e ocorre uma sobreposição de caracteres
4	Ativado com valor 1 quando há erro de formatação na recepção — número inadequado de bits de início e de parada
3	Ativado quando um sinal é recebido na linha CTS
2	Ativado quando um sinal é recebido na linha DCD
1	Ativado quando o registro de dados transmitidos está vazio
0	Ativado quando o registro de dados recebidos está cheio

Nosso segundo programa-exemplo utiliza um chip 6850 para receber, de um terminal, um string de caracteres, finalizado por um sinal de retorno do carro. Trata-se de programar adequadamente o chip e, então, verificar por meio de um loop se o registro de dados recebidos está cheio. Em caso afirmativo, removemos o byte de dados, que reinicializa o bit 0 no registro de status. Repete-se o processo até que o caractere recebido seja o de retorno do carro (13, no código ASCII). Devem-se ignorar quaisquer erros de transmissão, embora seja preciso checá-los. Para isso, mascara-se o conteúdo do registro de status para comprovar a ativação de algum bit indicador de erro. Utilizamos um protocolo bastante comum: 8 bits de dados, sem paridade, 2 bits de parada e um divisor de velocidade de clock (por 16). A primeira sub-rotina programa o chip, a segunda recebe os dados.

MÁQUINA DE DESENHO

As tartarugas, dispositivos robóticos relativamente baratos, ajudam os iniciantes a compreender os fundamentos da programação. Vamos ver os recursos de um versátil equipamento: o plotter Penman.

O BBC Micro está entre os computadores preferidos pelos pedagogos, podendo controlar inúmeros periféricos educativos — plotters, mouses, tartarugas. A grande vantagem do plotter Penman é reunir esses três dispositivos.

O conjunto compõe-se da unidade de controle, plotter móvel, fonte de energia, cabo de conexão RS232 e software correspondente. Quando fora de uso, a unidade de controle e o plotter ajustam-se, formando um conjunto compacto, de apenas 55 x 128 x 335 mm.

Para remover o plotter de seu invólucro, basta pressionar o grampo situado sob ele, retirá-lo e desenrolar o cabo conector paralelo. Ficam visíveis então três buracos, dentro dos quais se introduzem as canetas, e um orifício central, que acomoda uma outra caneta para desenhos do tipo executado com tartaruga. As canetas — hidrográficas — medem 40 mm; quando em posição, permanecem acima do papel por meio de molas. O comando para desenhar puxa a mola para baixo por ação de um eletroímã, e a ponta da caneta toca o papel.

Três rodas na parte de baixo movimentam o dispositivo. Duas delas, metálicas, respondem pela tração e a terceira, uma pequena roda de plástico que gira livre, apenas proporciona equilíbrio. As rodas metálicas são embutidas num invólucro áspero, que confere atrito adicional quando o aparelho se move sobre o papel. À frente de cada roda, um elemento fotossensor detecta a margem do papel ao registrar a diferença de brilho entre este e o material de baixo.

Dentro do plotter, três eletroímãs, conectados a pequenas alavancas, controlam, cada um, o movimento vertical de uma caneta. Ao lado, dois motores elétricos acionam independentemente as rodas. O plotter utiliza motores comuns, e não servomotores. Isso permite o desenho de curvas contínuas, embora exija um controle de software bem mais sofisticado, pois requer variação nas voltagens aplicadas aos motores, em vez de apenas uma série de pulsos.

Finalmente, um painel de circuito distribui a energia elétrica e decodifica os sinais provenientes da unidade de controle, enviando-os aos eletroímãs ou aos motores. Em cada eixo dos motores há um disco estroboscópico ladeado por dois diodos detectores de luz.

Esse disco gira em conjunto com o motor, gerando pulsos de luz nos diodos. Estes, por sua vez, informam à lógica interna do painel de circuitos a rotação do motor para que o computador possa calcular sua posição. Isso se mostra particularmente útil quando o Penman é usado como mouse, pois o computador precisa saber em que sentido e a que rotação as rodas estão girando, a fim de deslocar o cursor na tela.

O painel com o circuito impresso, dentro da unidade de controle, tem três chips principais. O primeiro, um microprocessador 6303 (desenvolvido a partir do Motorola 6800), possibilita a configuração de inúmeros sistemas operacionais e incorpora uma RAM de "rascunho" que o capacita a armazenar a posição do plotter. O segundo chip é uma interface periférica 6821; e

Personalidade dividida
O plotter Penman consiste num módulo de controle, conectado a um computador através de uma interface RS232C, e numa tartaruga. O cabo paralelo que liga a tartaruga ao módulo é bidirecional (ambos trocam informações). Enquanto a caixa de controle envia instruções à tartaruga para mover-se até determinada posição, esta, por sua vez, informa aos circuitos de comando qual a sua localização.

o terceiro, uma EPROM com os programas demonstrativos que o Penman executará se o cabo RS232C não estiver conectado.

Versatilidade operacional

O plotter Penman permite que o usuário escolha entre vários modos diferentes de utilização. No modo "emulador de terminal", o dispositivo é controlado por meio do teclado. Para selecionar esse modo, carrega-se o programa Pentlk diretamente ou, então, usando o robô Penman como mouse, a partir do menu principal. Carregado o software, enviam-se comandos às unidades de controle através da saída RS232C, sob a forma de códigos ASCII.

Para inicializar o Penman, digita-se PRINT 'I'. Ele receberá o sinal e determinará qual das três possíveis velocidades de transmissão, em bauds (bits/segundo), está sendo usada: 300, 1.200 ou 9.600. Feito isso, digita-se o comando H (home, "lar"), e tem início uma seqüência de movimentos que levam o robô ao canto inferior esquerdo da página.

De seu ponto de partida, o Penman tenta encontrar a margem da página usando seus fotossensores. Logo que a margem é encontrada, ele gira 90° e efetua a mesma ação ao longo da margem esquerda. Para garantir que o contraste entre o papel e a base seja captado pelo plotter, o conjunto inclui uma folha de papel preto, sobre a qual se apóia o papel de desenho. Depois que o Penman é colocado em "home", na margem inferior esquerda, ele estabelece sua posição a 5 cm de cada margem do papel (distância entre sua frente e o orifício central).

Executa-se a aplicação Pentlk no modo direto (um comando por vez) ou por intermédio de comandos reunidos em programas. Estes, por sua vez, são carregados (com LOAD), gravados em discos (com SAVE) ou executados (com RUN). O movimento no modo "emulador de terminal" baseia-se em coordenadas cartesianas — ou seja, o papel é considerado pelo robô como matriz.

Desse modo, quando recebe uma instrução para ir ao ponto (500,500), ele se move até essas coordenadas em vez de se mover quinhentos pas-

sos em ambas as direções. Uma folha de papel A4 possui um máximo de 2.100 coordenadas no eixo x e 2.970 no eixo y.

Os comandos podem ser introduzidos nos modos absoluto ou relativo, conforme a instrução MOVE seja prefixada por um A ou por um R. Controlam-se as canetas por intermédio de comandos semelhantes aos da linguagem LOGO: P (pen) para selecionar uma delas, U (up) para levantá-la e D (down) para baixá-la.

O plotter Penman também executa, no modo direto, movimentos gráficos do tipo tartaruga. São um pouco mais complexos que os movimentos cartesianos, visto que a distância a ser percorrida deve entrar no micro em notação hexadecimal prefixada por um $.

Isso porque os gráficos do tipo tartaruga se sobrepõem ao software normal que interpreta os movimentos cartesianos, indo diretamente aos bits nos endereços de controle, no interior do computador. Todavia, essa não é a única maneira de executar gráficos tipo tartaruga com o plotter Penman: o software aplicativo contém programas que permitem o controle do robô por meio da linguagem LOGO.

Controle em modo robótico

Um sistema semelhante para controle direto do Penman a partir do registro de dados do conector de expansões dirige o plotter em modo robótico.

Cada bit do registro controla um aspecto diferente dos movimentos do robô: os bits 0 e 1 e os bits 2 e 3 controlam, respectivamente, os motores direito e esquerdo; os bits 4 e 5 executam funções gerais dos motores, como ligá-los e desligá-los, e o bit 6 controla os movimentos da caneta, para cima e para baixo. O Penman dispõe ainda do recurso de transmitir ao registro de dados algumas informações a respeito de si mesmo, tais como erros de posicionamento e o aviso de que os fotossensores detectaram ou não a margem da página.

O plotter Penman aceita comandos de texto, que, tanto quanto suas aplicações, muito se assemelham aos utilizados nas impressoras/plotters disponíveis para grande número de microcomputadores pessoais. De fato, os caracteres de cada impressora apresentam notáveis semelhanças, podendo variar em tamanho de 1 a 127 mm de altura e ser impressos nos quatro sentidos. Um incremento útil do Penman, inexistente nas impressoras/plotters convencionais, é sua capacidade de inclinar o texto.

Instrumento pedagógico

O manual que acompanha o plotter apresenta uma compilação de todos os comandos disponíveis, com algumas explicações sobre a maneira como são implementados. Há também uma explanação extremamente minuciosa sobre a organização e o funcionamento do hardware. Talvez seja um tanto avançado para principiantes, podendo transmitir a idéia de que o plotter Penman se destina somente aos usuários que já dominam por completo o funcionamento de seus microcomputadores.

Em resumo, pode-se considerar o plotter Penman um avanço a mais no desenvolvimento de uma tartaruga/plotter, capaz de ser usada para finalidades múltiplas. Quando corretamente utilizado, sua resolução gráfica aproxima-se à dos plotters convencionais, encontrados nos estúdios de desenho comercial.

No entanto, o software existente impede que ele se torne um equipamento profissional. Ademais, seus programas objetivam, antes de tudo, a área educacional: espera-se que os usuários escrevam seus próprios programas para acionar o dispositivo. Isso os levará a aprender, pouco a pouco, os princípios do software robótico. Alguém que vise a finalidades comerciais certamente não se interessará por esses detalhes mais sutis da programação, exigindo, em vez disso, um aparelho que seja fácil de usar e resolva com eficiência seus problemas.

PLOTTER PENMAN

DIMENSÕES

335 x 128 x 55 mm.

INTERFACE

RS232C, que permite conexão a qualquer computador compatível com a RS232.

RESOLUÇÃO

Tamanho do texto: 1 a 127 mm; resolução gráfica: 0,1 mm.

DOCUMENTAÇÃO

O manual fornece numerosos detalhes técnicos, permitindo aos usuários já avançados desenvolver todo o potencial do Penman. Mas um principiante pode sentir falta de maior número de informações em nível elementar.

O controle dos "passos"

Para que o sistema de controle "saiba" a posição do plotter um mecanismo de retroalimentação no interior do robô informa o número de "passos" que foram dados. Trata-se de um disco de metal provido com numerosas fendas, ligado ao eixo de cada uma das rodas. Montado no painel do circuito, há um dispositivo que emite e recebe luz. À medida que o disco gira, produz-se um efeito estroboscópico entre o emissor e o receptor, na forma de uma série de pulsos. Pelo número de pulsos, os "passos" podem então ser calculados.

TEXTOS EM TELA

O processamento de texto, maior área de aplicações da microcomputação profissional, conquista atualmente os usuários domésticos. Quase todos escolhem o processador WordStar, recordista mundial de vendas.

O WordStar é um processador de texto orientado por menus, com formatação "transparente", ou seja, invisível para o usuário. A exemplo dos demais processadores, pode gravar documentos — desde uma única linha até páginas inteiras — em fitas ou discos e incorporá-los a outros textos. Pode localizar e substituir uma palavra ou expressão quantas vezes aparecer num documento, sendo capaz de alinhar o texto à esquerda, à direita ou de justificá-lo, isto é, variar o espaçamento entre as palavras para que todas as linhas tenham o mesmo tamanho.

Quase todos os softwares processadores de texto existentes no mercado possuem esses recursos. Contudo, na hora de adquirir um deles, a maioria dos usuários escolhe o WordStar. Vamos ver o porquê dessa preferência. O seu sistema operacional — o CP/M — controla a manipulação dos arquivos e as tarefas de manutenção. Acionando-se a tecla [Control], o computador diferencia entre comandos e caracteres. O usuário pode determinar que o menu de opções permaneça exposto numa seção de dez linhas, reservada para essa finalidade na parte superior do vídeo. Os comandos que definem a forma com que o texto aparece — tanto ao ser digitado, quanto no momento da impressão — são apresentados na área principal da tela, sem que sejam reproduzidos pela impressora.

Uma vez carregado no computador, o WordStar apresenta um menu inicial (ou de comandos), que permite a escolha da função desejada. Por meio dele, o usuário cria ou edita arquivos, na forma de documentos ou na de não documentos (usados, por exemplo, para conter as listas de nomes e endereços para mala direta); incorpora num único arquivo vários outros; seleciona a unidade de disco a ser utilizada; apaga, copia ou renomeia um arquivo; interrompe o programa e passa o controle ao sistema operacional; ou define o nível de auxílio necessário.

Nesse menu inicial, a escolha de cada opção é feita pelo acionamento de uma única tecla, dispensando-se o uso de [Control]. A tela mostra, então, o menu específico da função selecionada. A última das opções (definição do nível de auxílio) busca satisfazer tanto aos usuários novatos quanto aos experientes. Ela permite determinar como serão mostrados na tela os menus de comandos: quanto mais baixo o nível de auxílio requerido, menor o número de linhas reservadas para o menu e, conseqüentemente, maior o de linhas disponíveis para mostrar o texto. A maioria dos comandos é mnemônica — na medida em que o podem ser com apenas um caractere. Assim, a tecla [D] cria um arquivo de documentos, enquanto a tecla [N] cria um arquivo de não documentos.

A primeira informação solicitada pelo sistema é o nome do arquivo a ser criado. Depois, o sistema passa para o menu de operação. Este consiste num índice de funções, como, por exemplo, deslocamento do cursor, inserção ou eliminação de textos e lembretes dos códigos de controle para acessar outros menus. O WordStar não pressupõe que o computador no qual seja rodado disponha de setas para movimentação do cursor; o movimento de um caractere para o seguinte é definido pela tecla [Control], em conjunção com as teclas [E], [S], [D] e [X], que

Menu inicial

Menu inicial
Apresenta as funções básicas disponíveis. O usuário escolhe então entre criar ou editar um arquivo, seja do tipo documento — em que utiliza a justificação de linhas, a transferência de palavras para a linha seguinte e outros parâmetros de formatação padronizados —, seja do tipo não documento. Pode ainda copiar, renomear ou apagar um arquivo. Em resumo, essa parte do programa trata das tarefas de manutenção.

Menu de auxílio

Menu de auxílio
Acessado pelo código de controle J, fornece uma série de explicações sobre o modo de operação de partes específicas do software. Acrescentando ao código J um caractere suplementar — H, B, F, D etc. —, obtemos uma explicação completa a respeito daquelas seções do WordStar.

formam um "diamante" no teclado (para cima, para baixo, à esquerda, à direita). Portanto, os computadores com teclas de setas para controle de cursor têm essas funções duplicadas ao rodar o WordStar. Neste, esse controle é bastante versátil, regulando o seu deslocamento por caractere, palavra, linha ou parágrafo.

O mesmo princípio foi aplicado na função de eliminação. Caracteres isolados podem ser suprimidos, tanto no ponto em que está o cursor, quanto imediatamente à sua direita. Também é possível apagar palavras inteiras, à direita da posição do cursor, ou eliminar linhas.

As funções de controle também permitem deslocar o texto verticalmente na tela, sem mudanças no posicionamento do cursor. A utilidade desse recurso é evidente, caso seja preciso rever partes anteriores do texto.

Por último, o usuário tem a possibilidade de optar entre inserir um caractere na posição do cursor ou substituí-lo por outro, introduzido por meio do teclado. Essa função é do tipo liga/desliga, ou seja, o acionamento da tecla apropriada ([V], no caso) transforma o modo de inserção no de substituição. Este permanece ativado até que [V] seja pressionada de novo, juntamente com a tecla [Control]. O software controla automaticamente a paginação — pela qual se define o número de linhas por página — e, quando cada linha chega ao final, transfere as palavras para a seguinte.

Menu de comandos rápidos

Essas funções cobrem praticamente todos os aspectos da criação e edição de documentos. No entanto, o WordStar oferece recursos suplementares. Por meio deles é possível, por exemplo, centralizar títulos, sublinhar palavras e linhas, usar caracteres em negrito, e imprimir sobrescritos e subscritos (m^2 e H_2O, por exemplo). Todos esses recursos estão disponíveis no momento da criação do texto ou mais tarde, para se processarem as correções.

A função de "busca e substituição" é acessível por meio do menu Q (quick, "rápido"). Em primeiro lugar, o usuário digita os caracteres correspondentes à palavra ou frase a ser pesquisada. Depois, ele indica se deseja encontrar uma, várias, ou todas as ocorrências da palavra ou frase digitada. Decide, a seguir, se elas devem ser apenas localizadas ou substituídas por outras palavras ou frases. Realiza-se a substituição sem qualquer instrução adicional.

De modo semelhante, é possível copiar ou transferir blocos de texto de um lugar para outro, num documento, através das funções do menu K (block, "bloco"). Os três outros menus suplementares — J, O e P — abrangem, respectivamente, a explicação de códigos de controle, a formatação dos textos na tela e o modo em que estes devem ser impressos.

As características do WordStar permitem que ele seja executado numa grande variedade de computadores, desde que rodem no sistema operacional CP/M e, de preferência, aceitem 80 colunas na tela e letras minúsculas. Muitas dessas máquinas vêm equipadas com teclas de função programáveis. O IBM PC, por exemplo, tem dez delas. Cada uma pode ser programada para representar uma série de caracteres de controle, uma palavra ou uma frase empregada com mais freqüência. Dessa maneira, fica ainda mais fácil a sua inserção no texto.

O WordStar no Brasil

Os usuários brasileiros do WordStar sempre se ressentiram das dificuldades da acentuação. Nessa mesma palavra, para se escrever a última sílaba, digitava-se a letra c, depois um código de superposição e uma vírgula; tudo isso para "fazer" a cedilha. Escrevia-se então a letra a e repetia-se o código; a seguir vinham o til e a letra o. Todos os caracteres, das palavras e dos códigos, apareciam na tela.

Menu de blocos

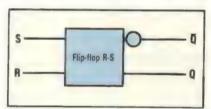
Em 1984, porém, a BraSoft lançou uma versão "acentuada" do WordStar, na qual os caracteres aparecem na tela tal como serão impressos. O programa se destina a micros de 16 bits (IBM PC e compatíveis), podendo também empreender a separação silábica em português. Vem acompanhado de um corretor ortográfico, um programa que funciona acoplado ao WordStar e que abrange cerca de 20.000 verbetes em inglês. Os fabricantes empenham-se na tradução para o português desse "dicionário eletrônico".

Menu de comandos rápidos
Permite ao usuário posicionar o cursor no início ou no fim da linha, da tela ou do documento, bem como utilizar as funções de busca e substituição. É acessado pelo código de controle Q, primeira letra da expressão inglesa quick command menu ("menu de comandos rápidos").

Menu de blocos
O menu K — denominado a partir da última letra da palavra block, "bloco" — controla a gravação e leitura de arquivos, o deslocamento e a cópia de blocos de texto. Ou seja, duplica algumas rotinas de manutenção do próprio software, acessíveis pelo menu inicial: gravação, eliminação, cópia e junção de arquivos.

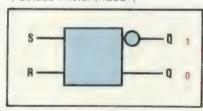
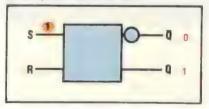
CIRCUITOS FLIP-FLOP

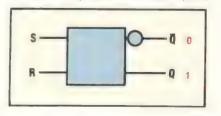
Os circuitos seqüenciais são capazes de produzir um sinal de saída contínuo, em resposta a um só pulso de entrada. Nesta matéria examinaremos o desenho e a função de um circuito assim, o flip-flop R-S.

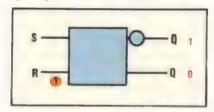
Há vários tipos de flip-flop, todos baseados no mesmo princípio. O flip-flop R-S possui duas linhas de entrada e duas de saída (Q e $\overline{Q}$):

Desenha-se o circuito de forma que as linhas de saída estejam sempre em oposição:

se Q = 1, então $\overline{Q}$ = 0 (estado SET);
se Q = 0, então $\overline{Q}$ = 1 (estado RESET).

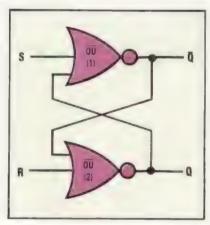
Supondo que o flip-flop comece no estado RESET, um pulso na linha S fará com que o circuito "salte" (flip) para o estado SET.

1) Estado inicial (RESET)

2) Um pulso na linha SET

Quando cessa o pulso de entrada da linha S, o circuito permanece no estado estável SET.

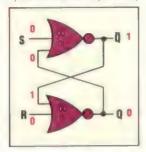
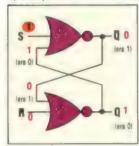
3) O circuito permanece estável (SET)

Um pulso enviado por R faz o circuito saltar (flop) para seu estado RESET inicial.

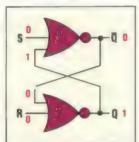
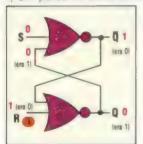
4) Um pulso na linha RESET

Descrita a função de um flip-flop R-S, passamos a examinar mais de perto os elementos lógicos do circuito.

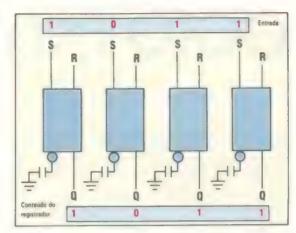
O circuito flip-flop R-S

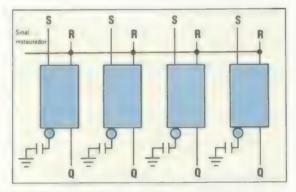
Podemos construir um flip-flop R-S mediante a ligação de duas portas E ou duas portas OU (caso deste exemplo), de modo tal que a saída de cada uma forneça uma das entradas para a outra. É esse "retorno em loop" dos sinais lógicos que permite a capacidade de "memória" do flip-flop:

Vamos acompanhar as funções SET e RESET e ver como a combinação de portas $\overline{OU}$ nos permite obtê-las. Supondo que o flip-flop inicie no estado RESET e que não haja pulsos de entrada, o estado do circuito será o estável (lembre-se de que a porta $\overline{OU}$ somente fornecerá uma saída de 1 se ambas as entradas forem 0).

Um pulso na linha S desestabilizará essa disposição, alterando a saída "não Q" ($\overline{Q}$) para 0. Isso afeta a entrada que "retornou em loop" para a segunda porta $\overline{OU}$ (2), fazendo com que a saída proveniente dessa porta (Q) se modifique para 1. Assim, se o pulso ainda está presente na primeira porta $\overline{OU}$ (1), as entradas para essa porta são ambas 1, e a saída, 0: o circuito alcança um estado estável, isto é, SET.

1) Estado inicial (RESET)

2) Um pulso na linha SET

Mesmo quando se retira o pulso da linha S, o circuito permanece estável. Quando se envia um pulso pela linha R, coloca-se o circuito num estado não estável. Após um processo semelhante ao já descrito, o circuito volta a um estado estável RESET.

3) O circuito permanece estável (SET)

4) Um pulso na linha RESET

Registradores

O microprocessador é em grande parte formado por uma série de registradores, como os acumuladores, os indexadores e os que processam instruções. A maioria dos registradores manipula palavras de 8 bits (números binários no limite de 0 a 255). Como têm de aceitar e armazenar informações binárias, esses registradores são constituídos por uma série de oito flip-flops. Para que um registrador de 4 bits armazene o número binário 1011, por exemplo, é necessário apenas fornecer o padrão binário às linhas S.

Observe que, nessa posição, a saída "não Q" não é utilizada. De acordo com o padrão binário aplicado às linhas S dos flip-flops, obtemos uma saída correspondente das linhas Q. Para substituir o primeiro número armazenado no registrador por outro, por exemplo, 0110, não basta apresentar esse novo padrão binário para as linhas S. Com tal procedimento, o número resultante armazenado no registrador seria 1111. Os 1 nas posições externas desse número são remanescentes do número anterior.

A solução será então restabelecer todos os flip-flops simultaneamente antes de armazenar o segundo número. Para tanto, convém conectá-los, permitindo restaurar o registrador por um único sinal.

No próximo artigo da série continuaremos examinando circuitos seqüenciais, inclusive o flip-flop tipo D e o flip-flop J-K.

Exercício 7

1) Por que o circuito flip-flop é também chamado "biestável"?

2) Quando um computador é ligado, um flip-flop está neste estado:

$Q=0, \overline{Q}=0, S=0, R=0$

a) Esse é um estado estável?

b) Se não for, para qual estado o flip-flop mudará?

c) Ele pode passar para um estado que não o da resposta anterior?

d) Ao ligar um computador, qual o processo para garantir que todos os registradores estejam num estado previsível?

GOTO END

Finalizando esta série sobre o PASCAL, discutiremos a necessidade fundamental de descrever de forma adequada os dados, além de mencionarmos outros aspectos não examinados anteriormente.

O criador do PASCAL, Niklaus Wirth, intitulou um de seus livros *Data structures + algorithms = = programs (Estrutura de dados + algoritmos = programas)*, o que reflete a importância da descrição de dados na formulação dos algoritmos que processam esses dados. Se usarmos uma matriz de CHARS para representar um string de caracteres, por exemplo, as funções e procedimentos necessários para processá-los (encontrar seu tamanho, concatená-los etc.) serão muito diferentes dos necessários no caso de se usar uma lista ligada.

Suponha, por exemplo, a tarefa simples de determinar o tamanho de um string. Lembre-se de que preenchemos todos os elementos não utilizados da matriz com caracteres NUL em código ASCII, CHR (0), em se tratando de matriz, ou encerramos a lista dinâmica com o valor indicador NIL. A versão da matriz pareceria bastante simples:

```
FUNCTION TAM (S : STRING) : CARDINAL;
  VAR
    N : 0 .. MAXSTRING;
    ACHOU : BOOLEAN;
```

Programa TreeSorter

Utiliza alguns dos procedimentos do programa ListaCirc, assinalados no ponto apropriado. Dados envolvendo nomes, totais de saldo devedor e quaisquer outros campos que você queira acrescentar serão fornecidos por meio do teclado e inseridos em ordem alfabética crescente numa "árvore binária".

Cada nó da árvore dispõe de dois POINTERS para ligação a itens "menores" ou "maiores". Obtém-se a transversal da árvore pela comparação dos novos dados com o campo NOME em cada nó, e pela tomada do campo LIGA apropriado (ABAIXO ou ACIMA). Quando encontramos um nó vazio (isto é, como um desvio de valor NIL), inserimos o item. Gravam-se os dados no arquivo, portanto, mediante um simples procedimento recorrente.

Esses exemplos servem como base para um gerenciador de banco de dados bem organizado — uma seqüência de programas adaptada a seu próprio uso, por exemplo. Apenas uma advertência: se você ler um arquivo classificado para uma árvore binária, todas as inserções estarão no mesmo desvio, e a árvore se tornará uma lista de ligação simples.

Para aplicações mais simples, a lista circular é provavelmente a mais versátil. Não há qualquer indicador NIL nelas, e isso evitará um teste duplo em cada comparação, ao passo que a árvore binária terá NILs (NUMERO DE NOS + 1).

```
PROGRAM TREESORTER (INPUT OUTPUT ARQDADOS)

CONST
      NOMARQ       = DADOS
      MAXSTRING    = 25

TYPE
    CARDINAL       = 0   MAXINT.
    TAMSTRING      = 1   MAXSTRING
    STRING         = PACKED ARRAY [TAMSTRING] OF CHAR
    REGISTRO       = RECORD
                          NOME          STRING
                          OUTROS CAMPOS
                          DEBITO        CARDINAL
                     END  REGISTRO
    TIPOARQ        = FILE OF REGISTRO
    ARVORE         =  NO
    NO             = RECORD
                          ITEM          REGISTRO
                          ABAIXO
                          ACIMA         ARVORE
                     END  NO

VAR
    DADO             REGISTRO
    ARQDADOS         TIPOARQ
    TRONCO           ARVORE

INCLUDE  UTILIT  SRC   ARQUIVO FONTE CONTENDO
                      SALTABR LERCARD LERLINHA  -
                       VEJA O PROGRAMA LISTACIRC

PROCEDURE LERQUANT (VAR QUANT   CARDINAL)
        ** VALIDA UM VALOR CARDINAL **
  VAR
      OK                BOOLEAN
  BEGIN
    REPEAT
      REPEAT
        WRITE ('QUANTIDADE?'  20)

        IF EOLN (INPUT) THEN
          READLN (INPUT)

        SALTABR (INPUT)
      UNTIL NOT EOLN (INPUT)

      LERCARD (INPUT QUANT OK)

      IF NOT OK THEN
        WRITELN (' — ERRO --'  20  REENTRE)
    UNTIL OK  EXIGE UM NUMERO VALIDO

  END  LERQUANT

PROCEDURE INSERE (VAR NOVONO   ARVORE
        COLOCA NA ARVORE  DADO   REGISTRO)
  BEGIN
    NEW (NOVONO).

    WITH NOVONO   DO
```

```
BEGIN
  N := 0;
  ACHOU := FALSE;
  REPEAT
    N := N + 1;
    ACHOU := S[N]=CHR(0)
  UNTIL ACHOU OR (N = MAXSTRING);
  IF ACHOU
    THEN
      TAM := N-1
    ELSE
      TAM := MAXSTRING
END; {TAM}
```

Há aqui alguma possibilidade de erro e confusão. Por que precisamos da variável booleana local ACHOU? E por que temos de atribuir o valor N−1 a TAM? São problemas de pouca importância, mas a formulação de algoritmos mais complexos pode se tornar confusa e passível de muitos erros, se tivermos de recorrer a artifícios desse tipo.

O uso de "flags de bit" (que conhecemos como tipos booleanos) é um dos artifícios mais usados pelos programadores, mas freqüentemente apenas para atender às exigências do computador e não como uma parte natural do algoritmo. Deve-se usar a variável local N, e não TAM, já que esta é um identificador de função, e não uma variável. Identificadores de função do lado direito de uma instrução como

```
TAM := TAM + 1
```

tentariam fazer uma chamada recorrente à própria função TAM.

Compare isso com a função TAM necessária para strings que usam uma lista ligada. Com a representação dinâmica, lembre-se de que a definição TYPE de STRING é muito diferente e possibilita a criação de qualquer tamanho de string. Esse tipo de descrição de dados muitas vezes será definido de forma recorrente.

Recorrência

Muitos dos exemplos usados para ilustrar a recorrência poderiam ser igualmente bem expressos (se não melhor) por meio de algoritmos iterativos; mas examinemos a função TAM para o tipo string recorrente.

```
    BEGIN
      ITEM  = DADO
      ABAIXO  = NIL  AINDA SEM DADOS
      ACIMA  = NIL  LIGADOS A ESTE NO
    END

  END  INSERE

PROCEDURE DIRECAO (TRONCO  ARVORE
  ACHA POSICAO  DADO  REGISTRO)
  VAR
      TRTEMP        ARVORE
      MAIOR         BOOLEAN
  BEGIN
    WHILE TRONCO  < > NIL DO
      BEGIN  ENCONTRA UMA POSICAO VAZIA
        TRTEMP  = TRONCO
        MAIOR  = DADO  NOME  
                  TRTEMP  ITEM  NOME
        IF MAIOR
          THEN  DIRECIONA ACIMA
            TRONCO  = TRONCO  ACIMA
          ELSE  DIRECIONA ABAIXO
            TRONCO  = TRONCO  ABAIXO
    END

  INSERE (TRONCO DADO

  IF MAIOR
    THEN INSERE ACIMA
      TRTEMP  ACIMA  = TRONCO
    ELSE  INSERE ABAIXO
      TRTEMP  ABAIXO  = TRONCO

  END  DIRECAO

  PROCEDURE     PRINT (VAR F  TIPOARQ
  GRAVA DADOS NO ARQUIVO RAIZ  ARVORE)
  BEGIN
    IF RAIZ  < > NIL THEN
      WITH RAIZ   DO  RECORRENCIA
        BEGIN  ESCREVE NA ORDEM
          PRINT (F  ABAIXO)  ABAIXO PRIMEIRO
          WRITE (F  ITEM)  DEPOIS A RAIZ
          PRINT (F  ACIMA)  ACIMA POR ULTIMO
        END

  END  PRINT

PROCEDURE LIBERA (VAR RAIZ   ARVORE)
  LIBERA MEMORIA PARA USO POSTERIOR
```

```
  BEGIN
    IF RAIZ < > NIL THEN  ** RECORRENCIA **
      BEGIN  LIBERA DESVIOS PRIMEIRO
        LIBERA (RAIZ   ABAIXO)
        LIBERA (RAIZ   ACIMA)
        DISPOSE (RAIZ)  DEPOIS A RAIZ
      END

  END  LIBERA

BEGIN    TREESORTER -  PROGRAMA PRINCIPAL

  ASSIGN (ARQDADOS NOMARQ)
  PAGE (OUTPUT)
  WRITELN (' = = = CLASSIFICACAO EM ARVORE = = =    25)
  WRITELN
  WRITELN ( ENTRE COM DADO (SOBRENOME ANTES) )
  WRITE ( NOME? )
  LERLINHA (DADO  NOME)
  LERQUANT (DADO  DEBITO)
  INSERE (TRONCO DADO)  COMECA A ARVORE
  WRITE ( NOME ? )
  LERLINHA (DADO  NOME)

  WHILE DADO  NOME [1] < > CHR (0) DO
    BEGIN
        LERQUANT (DADO  DEBITO)
        DIRECAO (TRONCO DADO)
        WRITELN ( RETURN QUANDO ACABAR    40)
        WRITE ( NOME ? )
        LERLINHA (DADO  NOME)
    END
     OUTROS PROCESSOS  DEPOIS GRAVE O ARQUIVO
  WRITELN ('GRAVANDO EM      NOMARQ)
  REWRITE (ARQDADOS)
  PRINT (ARQDADOS TRONCO)
  WRITELN
  WRITELN ( DEBITO'   8.  NOME    MAXSTRING).
  WRITELN
  RESET (ARQDADOS)  LER O ARQUIVO
        E IMPRIMIR A LISTA CLASSIFICADORA
  WHILE NOT EOF (ARQDADOS) DO
    BEGIN  LE CADA REGISTRO
        READ (ARQDADOS DADO).
          MANDA DADOS P/ O VIDEO
        WITH DADO DO
          WRITELN (DEBITO   8.'   '. NOME)
    END

  LIBERA (TRONCO)  DISPOSE DA MEMORIA
    PARA OUTRO PROCESSAMENTO ETC
END
```

Criando um novo nó
O programa TreeSorter utiliza uma estrutura de árvore binária para armazenar dados classificados por um campo-chave. Os itens são inseridos nos desvios inferior ou superior, conforme a ordem alfabética do campo.
O procedimento INSERE cria um novo nó, enquanto DIRECAO determina a rota para o desvio NIL adequado.
O diagrama mostra uma árvore binária simples, onde cada nó mantém um único caractere (LET), e o modo de armazenamento de um string simples de minúsculas ('pascal'). A área em vermelho indica um estágio no desenvolvimento da árvore, tal como efetuado pelo procedimento INSERE.

```
FUNCTION TAM (S : STRING) : CARDINAL;
  BEGIN
    IF S = NIL
      THEN TAM := 0
      ELSE TAM := SUCC (TAM (S ^ PROX))
  END; TAM
```

O cabeçalho da lista é S, e a expressão S ^ PROX seleciona o campo indicador do registro seguinte da lista. Ou, então, ele pode ser considerado como uma lista que começa com o registro seguinte (PROXREG). Sempre que não encontramos um NIL, chamamos uma avaliação do tamanho de PROXREG e aumentamos com a função SUCC.

O exemplo anterior, bastante comum, permitirá escrever TAM sem o uso da recursividade. O código é maior e deve-se usar um contador local (como na versão da matriz) especificamente para evitar a recorrência.

```
FUNCTION TAM (S : STRING) : CARDINAL;
  VAR
    N : CARDINAL;
  BEGIN
    N := 0
    WHILE S < > NIL DO
      BEGIN
        N := N + 1
        S := S ^ PROX
      END
    TAM := N
  END; ;TAM
```

Muitos compiladores PASCAL admitem "diretivas", que são instruções para o compilador — e não declarações ou comentários. A única diretiva realmente necessária pelo padrão ISO é FORWARD. Caso dois procedimentos precisem chamar um ao outro, eles serão denominados "mutuamente recorrentes". Isso acontece muito raramente, mas coloca um problema: não podemos utilizar um objeto em PASCAL sem antes declará-lo ou defini-lo.

A solução será declarar somente o cabeçalho do subprograma, substituindo seu bloco pela diretiva FORWARD do compilador. Após a definição completa do outro módulo, o cabeçalho recebe sua forma abreviada (sem a lista de parâmetros), e o bloco é definido nesse ponto. Muitas vezes há outras diretivas para controlar as opções de controle durante a compilação, mas não deverão ser usadas com freqüência, se você quiser garantir a portabilidade. Isso significa que, além de um símbolo especial (geralmente $) aparecer como o primeiro caractere do comentário, as opções não portáteis podem não ser entendidas por um outro compilador.

Outras diferenças

Algumas implementações não padronizadas (especialmente a HiSoft) exigirão sintaxe um pouco diferente para as declarações indicadoras FORWARD. Há pouquíssimas "variações" desse tipo e nenhuma delas será encontrada em compiladores que seguem a definição ISO.

Uma descrição avançada de dados, muito importante no PASCAL e que não vimos, é a "variante". Para armazenar itens de diferentes tipos, utilizamos um registro com campos adequados. Mas, e se precisássemos tornar flexível a descrição de parte ou mesmo de todo o registro? De modo geral, poderíamos querer armazenar diferentes informações sobre as pessoas, dependendo, por exemplo, de serem ou não casadas. Um registro de variante possui sua "parte fixa" definida em primeiro lugar; depois, especifica-se a parte variável pela introdução de um seletor de variante (de qualquer tipo simples) e pelo uso das palavras reservadas CASE e OR. Por exemplo:

```
TYPE
  GENERO = (MASCULINO, FEMININO);
  VARIANTE = RECORD
    CAMPOS COMUNS
    CASE ESTCIV : BOOLEAN OF
      FALSE : ();
      TRUE : (DATACAS : STRING;
          CASE SEXO : GENERO OF
            MASCULINO : ();
            FEMININO : (ESPOSO : STRING))
      END; ;VARIANTE
```

Observe que as listas de campo vazias devem ainda estar entre parênteses. O espaço em um arquivo será fixo (de acordo com a maior variante), mas pode-se economizar memória com indicadores para variantes, como, por exemplo, NEW (P, TRUE, MASCULINO).

Os poucos pontos frágeis do PASCAL foram em grande parte eliminados pelos especialistas (e por Wirth, em MODULA-2). A linguagem é bastante poderosa (embora pequena), eficiente, destinada a usos gerais e fácil de aprender.

SORD

No competitivo mercado japonês, dominado por gigantescas corporações, a SORD, uma pequena empresa com algumas centenas de empregados, vem se destacando por seus inovadores projetos informáticos.

Takayoshii Shiina
Fundador e presidente da SORD

O curioso nome da SORD (combinação de SOftware e haRDware) é adequado, já que ela sempre dedicou atenção tanto ao desenvolvimento do software quanto do hardware.

A companhia teve começo um pouco vacilante. Em 1967, seu futuro presidente, Takayoshii Shiina, deixou a universidade para reorganizar o departamento de marketing da Rikei, um conglomerado industrial de porte médio. O fato de Shiina ter conseguido esse emprego chega a ser algo incomum, visto que, no Japão, os empregados de uma empresa raramente mudam de emprego, nele permanecendo por toda a vida. Segundo consta, já em sua primeira entrevista, ele declarou que, como planejava dirigir sua própria firma, trabalharia na Rikei apenas por alguns anos.

Em 1970, ele e um amigo estavam prontos para lançar o novo empreendimento e constituíram a SORD com um pequeno capital. Shiina, no entanto, continuou na Rikei até dezembro daquele ano. O primeiro produto fabricado pela SORD foi um testador lógico de baixo custo; além disso, a empresa obteve também alguns contratos para trabalhos de programação.

Em 1971, a SORD começava a envolver-se numa ampla gama de negócios, mas o início da produção de equipamentos deu-se dois anos depois, e no final de 1974 a SORD apresentou uma unidade de discos flexíveis que operava com uma interface por ela desenvolvida. Seguiu-se, logo, o SMP-80/20, um dos primeiros computadores do Japão baseados no processador Intel 8080.

O SMP-80/20 foi um produto de grande sucesso e logo multiplicaram-se as encomendas. Em 1977, visando à expansão de sua empresa, Shiina vendeu à Toppan, uma das maiores gráficas do Japão, 20% das ações sob seu controle. Essa injeção de capital permitiu que a SORD criasse considerável base de software para sua lista cada vez maior de computadores. Desse trabalho resultou, em 1981, o PIPS (Pan Information Processing System, "sistema de processamento multiinformacional").

Um dos primeiros softwares integrados, o PIPS estava muito à frente dos pacotes existentes na época, e ajudou a consolidar a posição da SORD no mercado. Esse pacote combina as funções de planilha financeira, processador de textos e banco de dados, de tal modo que até pessoas que nunca manejaram um computador aprendem a usá-lo em poucas horas.

O sucesso fenomenal do PIPS no mercado interno japonês só pode ser entendido com uma explicação sobre suas características na época. No Japão, era prática corriqueira a venda de computadores sem software de apoio. Embora comum na Europa e nos Estados Unidos, o software aplicativo do tipo não profissional — para contabilidade, faturamento etc. — era virtualmente desconhecido no país.

A carência de aplicativos no mercado japonês deixava um espaço enorme para a SORD explorar. A empresa começou a desenvolver a série M200 de computadores baseados no Z80 e, mais tarde, a série M23. O M5 foi introduzido para atender o mercado de jogos domésticos. Em meados da década de 80, foi comercializada a série M68 — que incorporava tanto o microprocessador Z80 como o Motorola 68000 — e os pequenos micros comerciais M243, além do computador M285, de 32 bits, que roda o software VAX-11 para aplicativos CAD (Computer-Aided Design, "projeto auxiliado por computador").

PIPS
Desenvolvido pela SORD com características únicas, este software de uso geral adapta-se à maioria das aplicações comerciais, da contabilidade aos gráficos promocionais. Demonstra a peculiar visão japonesa de marketing, pois só pode ser utilizado nos equipamentos SORD. Embora a eficiência do PIPS constitua um grande atrativo, sua especificidade limita bastante a demanda.

Nenhuma outra companhia no mundo produz uma linha tão variada de computadores, cada um com vasto software de apoio. Por outro lado, a SORD parece falhar ao ver seus produtos como "fins em si mesmos", isto é, em considerar suas máquinas como sistemas totalmente operacionais, completos, como hardware e software. Em geral, os usuários de um SORD não dispõem da opção de escolher outros produtos de software para sua máquina.

A companhia buscou corrigir essa situação fornecendo o sistema operacional SB-80 como opção para sua série M23 de computadores baseados no Z80. O SB-80 equivale ao sistema operacional CP/M 2.2, da Digital Research, e permite o uso do software relacionado ao CP/M nos computadores SORD. Outro passo da empresa nesse sentido foi tornar seus equipamentos compatíveis com o p-System, elaborado na Universidade da Califórnia, em San Diego (UCSD).

Para os usuários satisfeitos em limitar-se ao software da própria SORD, a companhia oferece vários BASICs potentes, seu próprio sistema operacional, o software integrado PIPS e um processador de textos.

Além de dar acesso ao CP/M e ao p-System, a SORD tende a seguir a trilha aberta por fabricantes de minicomputadores, que oferecem sistemas dependentes de seu próprio software. Enquanto isso, as indústrias de computação restantes descobriram que o caminho para melhorar as vendas consiste na padronização e na compatibilidade entre famílias de computadores. A SORD tem revelado lentidão em se adaptar a essa tendência. Continua a desenvolver projetos inovadores de hardware, mas, dizem os críticos, falha em não apoiar os usuários com documentação adequada para seus produtos.

Desde 1983 a SORD tem se esforçado em elaborar boa documentação para o PIPS e seu processador de textos. Mas, mesmo em 1985, faltava documentação mais aprofundada de seu hardware, e notava-se sua relutância em encorajar outros fabricantes de software a desenvolver produtos que rodassem em seus computadores.

Os próximos anos serão críticos para a SORD, que se defrontará com a IBM e com o poderio industrial dos maiores fabricantes de computadores do Japão. Seu presidente é um fervoroso defensor da iniciativa individual e da pequena empresa, e teve sucesso em fazer da SORD uma companhia internacional. Será ela capaz de manter-se no mundo da informática?

Opções SORD
As máquinas da SORD mais conhecidas são o computador pessoal M5 e o portátil comercial M23P. Este último impôs novos padrões para os portáteis ao usar discos flexíveis Sony e um visor de cristal líquido de 80 colunas.

TAL E QUAL

A transmissão de documentos via telefone difundiu-se no Brasil graças à IFAX 3021, um equipamento fac-símile produzido pela Itautec com base em tecnologia desenvolvida pela Canon japonesa.

Em 1979, a Xerox lançou no mercado americano sua telecopiadora 485, capaz de reproduzir em menos de 1 minuto uma carta comercial padrão enviada à distância. Totalmente automatizada e mais veloz que suas precursoras, era o quarto modelo de equipamento fac-símile lançado no mercado pela empresa.

Tida como introdutora do sistema fac-símile também no Brasil, devido à comercialização do modelo Telecópia 400, a Xerox dividia o mercado com a Ecodata (subsidiária da Cable and Wireless, da Inglaterra) e a NEC do Brasil (subsidiária da Nippon Electronic Corporation, do Japão). No entanto, em função do alto custo e da falta de interesse pelo produto, esse equipamento importado não obteve sucesso.

O princípio do funcionamento da telecopiadora fac-símile é o mesmo dos equipamentos de telefoto e radiofoto: transmite por impulsos sonoros via rádio ou telefone.

A Itautec (Itaú Tecnologia), acreditando nas vantagens das fac-símile em relação ao tradicional sistema de transmissão via telex, enviou sete engenheiros de seu Departamento de Projetos ao Japão para um estágio de dois meses na Canon Corporation. Essa indústria detém a tecnologia do modelo FAX 510, que passou a ser produzido pela Itautec como IFAX 3021.

O modelo escolhido para comercialização no Brasil apresenta a característica da compatibilidade com outras telecopiadoras fac-símile, dos grupos 3 e 2, e com as mais antigas, baseadas no modo de transmissão FM (americano e euro-

Nacionalização crescente
O projeto de progressiva nacionalização da IFAX 3021 foi aprovado pela SEI (Secretaria Especial de Informática) em outubro de 1984. Já no mês seguinte, o equipamento era apresentado ao público durante a Feira de Informática, no Rio de Janeiro.

Qualidade das imagens
Nesta folha de teste, vêem-se os diversos recursos de resolução gráfica da fac-símile. Textos, mapas, fórmulas e diagramas podem ser transmitidos com todos os detalhes.

peu). A classificação em grupos (segundo o tempo de transmissão de uma página tamanho A4) se deve à padronização pelo CCITT (Comitê Consultivo Internacional de Telegrafia e Telefonia), com sede em Genebra, Suíça. No grupo 1, o tempo de transmissão vai a 6 minutos; no 2, equivale a 3 minutos; e, no 3, fica entre 15 segundos e 1 minuto. A IFAX 3021, portanto, é um equipamento de alta velocidade do grupo 3, mas opera também no grupo 2. Selecionando automaticamente a velocidade apropriada à transmissão, ajusta-a ao aparelho receptor. Cada transmissão realizada registra o modo como se emitiu (G2 ou G3), o número do telefone, o código dos assinantes (receptor e transmissor) e o número de páginas enviadas.

Em caso de erro de transmissão, o painel emite um sinal luminoso. A IFAX imprime o tipo de erro cometido e o número da página em que ocorreu. Isso permite rápida localização e correção do problema.

A recepção automática 24 horas por dia, opcional, permite que um documento seja recebido mesmo na ausência do operador. Além da recepção automática, a IFAX 3021 conta com um programa de transmissão de documentos, que apenas são colocados na bandeja, ficando dispensada a presença do operador. O alimentador de papéis permite posicionar até cinco originais na bandeja de documentos, transmitindo-os automaticamente.

A IFAX 3021 tanto pode imprimir cópias para checar a qualidade da imagem, antes da transmissão, como para uso local. Há três opções de contrastes, sendo que, para reproduções mais detalhadas, há ainda o modo fino, que possibilita maior resolução de imagem, para legibilidade máxima em qualquer dos contrastes. Durante a recepção ou transmissão, pode-se acionar a tecla [Falar], alertando o outro lado para atender o telefone antes ou depois de finalizada a operação.

O papel, do tipo termossensível, vem acondicionado em bobinas de 100 m, não exigindo trocas constantes; e a máquina se encarrega de cortá-lo no tamanho apropriado. A largura da imagem é de até 280 mm.

Na maioria das fac-símile, os documentos impressos são recebidos em ordem decrescente, o que requer posterior ordenamento à mão. A IFAX 3021 posiciona as cópias recebidas na ordem correta, dispensando esse trabalho.

A instalação requer uma tomada de 110 V e um telefone. Completada a ligação telefônica para onde se vai transmitir o documento, ouve-se um sinal telefônico contínuo; em seguida, o operador digita a tecla [Início] para acionar o processamento.

A fac-símile transmite qualquer tipo de original, manuscrito ou datilografado, impressões digitais, recortes de jornal, páginas de livro e impressos em qualquer tipo de papel. Atende a todo segmento de mercado que depende de comunicação rápida de textos, diagramas, gráficos, figuras, tabelas, fórmulas e mapas. Uma vez que o equipamento opera em branco e preto, sem o uso de meio-tom (cinza), a reprodução de fotos se faz em alto contraste. A velocidade de transmissão varia de 2.400 a 9.600 bauds (bits por segundo).

O sistema de leitura se faz por um chip que transforma sinais ópticos em elétricos. A leitura e o armazenamento se fazem linha a linha, sucessivamente. Como o papel de impressão vem enrolado, a transmissão-recepção é contínua. O sensor de imagem divide a linha da varredura horizontal em elementos de imagem, de forma que cada linha possa ser lida a seu tempo.

A transmissão de dados compreende três tipos: TSL (Transmissão por Salto de Linha), TB (Transmissor de Buffer) e MH (Modified Huffman). Como o tempo de transmissão depende da quantidade de informações na página, o TSL ignora as entrelinhas e emite um comando para saltá-las. O MH permite economizar, comprimindo os dados por meio de códigos de redundâncias, o que também abrevia o tempo de transmissão.

Ainda recém-comercializada, a IFAX 3021 foi adquirida pela Empresa Brasileira de Correios e Telégrafos para atender à rede de agências que operam o post-grama. Em 1985, esse serviço se estendia a 45 cidades em território nacional e a 24 países. Para as cidades brasileiras, o prazo de entrega do documento é de 1 hora e 30 minutos. Para o exterior, chega ao destino em 12 horas, quando expedido antes do meio-dia (horário nacional). A transmissão é feita entre correios, e o custo das tarifas nacionais varia conforme a cidade.

Simplicidade de operação
Da esquerda para a direita, vemos as diversas fases de funcionamento da IFAX 3021: primeiro, os documentos originais são colocados na bandeja voltados para baixo. Depois, chama-se pelo telefone o número da máquina receptora. Basta então acionar a tecla [Início] após o sinal de recepção. Esta é automática e não depende da presença de um operador.

CONTROLE AUDITIVO

Variações

Este programa foi escrito para o TK 90X. As instruções PRINT pressupõem o uso de um vídeo com 32 colunas, devendo ser alteradas se você quiser adaptá-las para um micro com tela de quarenta colunas.

Os comandos SOUND, RAND e RND são específicos do BASIC do TK 90X

A seguir apresentamos as alterações necessárias para adaptar o programa ao CP 400 Color.

- Altere a sintaxe das instruções PRINT, deixando-as coerentes com o BASIC do CP 400.
- Elimine a linha 20.
- Altere a linha 190 para:
  190 PRINT D: IF D 254
  THEN D = 254
- Altere a linha 200 para:
  200 SOUND (255-D),1

A maioria dos jogos eletrônicos utiliza os efeitos visuais do microcomputador. Apresentamos aqui um que se baseia na capacidade do micro de produzir diferentes sinais sonoros.

Os jogos do tipo fliperama usam sons para complementar seus vibrantes efeitos visuais, aumentando o realismo e a excitação. Raramente a sonorização constitui o tema central de um jogo. Contudo, da mesma forma que fazemos valer nosso sentido de visão, nada impede que usemos a audição como sua base.

Sinal de Alerta utiliza de forma interessante os recursos sonoros. A tarefa, que à primeira vista parece fácil, transforma-se em fascinante desafio, pois exige um sentido pouco usado nos jogos de ação, com seus movimentos rápidos e multicoloridos.

Você comanda uma nave espacial cujo combustível está terminando. Para sobreviver, precisa chegar a uma estação reabastecedora nas proximidades. Mas seu sistema de orientação por computador está com defeito e você não dispõe de indicação visual para se guiar na aterrissagem. Sua única referência é um sinal sonoro, emitido pela estação, que vai se tornando mais agudo à medida que prossegue a aproximação. Portanto, ouça com atenção e reaja rapidamente, acionando as teclas de controle: [I], para cima; [M], para baixo; [K], para a direita; e [J], para a esquerda.

Tal como sucede nas naves verdadeiras, uma vez impulsionada, ela prosseguirá na mesma direção até que você compense o empuxo inicial com um comando oposto. Se utilizar duas vezes [I] para subir rapidamente, terá de teclar [M] duas vezes a fim de parar. Esse fator torna o jogo muito mais difícil do que parece à primeira vista.

Você vai precisar de muita prática para conseguir pousar apenas com os sinais sonoros. Contudo, acrescentando algumas linhas ao programa, será incluída uma indicação visual na tela, que mostrará a posição da nave em relação à estação. Ela pode mostrar a distância que o separa da estação (a variável D) ou indicar que teclas devem ser pressionadas — por exemplo, se SGN (p − x) = − 1, então você deve virar à esquerda. Essas dicas facilitariam bastante o jogo.

Após experimentar nosso programa, talvez você queira criar seus próprios jogos. Usar os sons para localizar objetos ou desviar-se deles é um desafio que oferece muitas possibilidades ao programador. Que tal um jogo para submarinos com sonar, ou um campo minado que você tem de percorrer utilizando um detector que emite alarmas sonoros?

Será vantajoso programar esses jogos para micros com maiores recursos sonoros, como o TK 90X ou o CP 400 Color. Podem-se utilizar vários sons ao mesmo tempo, cada um com seu significado; o volume, programável em algumas máquinas, seria usado como recurso extra.

```
10  REM ** SINAL DE ALERTA **
       *** PARA O TK 90X ***
20  RAND
30  PRINT "MOVA SUA NAVE PARA
    ESTACAO REABASTECEDORA ANTES
    QUE O COMBUSTIVEL TERMINE"
40  PRINT : PRINT "A ESTACAO EMITE UM
    SINAL SONORO. QUANTO MAIS
    PERTO VOCÊ CHEGAR, MAIS AGUDO
    SERA O SINAL": PRINT "I (SOBE)", "M
    (DESCE)","J (ESQUERDA)","K
    (DIREITA)
50  PRINT: PRINT "PARA INICIAR DIGITE
    UMA TECLA"
70  LET A$ = INKEY$: IF A$ = " " THEN
    GOTO 70
80  LET P = INT (RND * 100) + 1: LET Q =
    INT (RND * 100) + 1: LET F =
    3 * (P + Q)/2: LET X = 0: LET Y = 0
90  LET XD = 0: YD = 0
100 PRINT "COMBUSTIVEL: ";F;TAB (0);"
    DISTANCIA: ";
110 IF (ABS (P  X) < 2 AND ABS (Q  Y) < 2)
    THEN PRINT 0: PRINT "VOCE CHEGOU
    A BASE COM";F;"GALOES DE
    COMBUSTIVEL": STOP
120 LET A$ = INKEY$: IF A$ = "" THEN
    GOTO 120
130 IF (AS$ = "I" AND YD < 3) THEN LET
    YD = YD + 1
140 IE (AS$ = "M" AND YD > (3) THEN LET
    YD = YD  1
150 IF (A$ = "J" AND YD > - 3) THEN LET
    XD = XD - 1
160 IF (A$ = "K" AND YD < 3) THEN LET
    XD = XD + 1
170 LET X = X + XD: LET Y = Y + YD
180 LET D = SQR ((P - X) * (P - X) +
    (Q  Y) * (Q - Y))
190 PRINT D: IF D > 129 THEN LET
    D = D  129
200 SOUND ,4,(69  D)
210 LET F = F  ABS (XD) - ABS (YD): IF
    F > 0 THEN GOTO 90
230 PRINT TAB (0);"
    ><><>.  .  .  .  .   .CRASH
    <><><><><><        ": FOR G = 1
    TO 50: PRINT AT 21,0;"
                  ";AT 21,0;"    COMBUSTIVEL
    ESGOTADO": NEXT G
240 PRINT : PRINT "VOCE FICOU PERDIDO
    NO ESPACO A       ";D;" UNIDADES
    ASTRONOMICAS", "DA BASE"
260 STOP
```

C

CRIAÇÃO DE REGISTROS

Muitos sistemas gerenciadores de bancos de dados possuem uma linguagem embutida. Procedimentos escritos "sob medida" nessas linguagens aumentam ainda mais a eficiência dos SGBDs.

Um banco de dados faz bem mais do que apenas organizar os registros de um arquivo. É possível ter acesso a registros por meio de comandos simples, mas os melhores sistemas gerenciadores de bancos de dados (SGBDs) permitem ao usuário escrever procedimentos utilizando suas próprias linguagens incorporadas. Isso amplia bastante a eficiência e a utilidade de um banco de dados.

Para examinar o uso em bancos de dados de procedimentos escritos pelo usuário, devemos primeiro rever a maneira pela qual a maioria dos SGBDs utiliza comandos — SEARCH, LOCATE, DISPLAY, NEXT, LAST e assim por diante. Esses comandos são processados pelo SGBD para descobrir o que o usuário deseja. O sistema gerenciador abre o arquivo do banco de dados e o percorre do primeiro ao último registro, selecionando no caminho os registros que satisfaçam os requisitos. Costuma-se dizer que esses comandos pertencem a uma "linguagem de consulta".

Em contraste com os comandos diretos ou de consulta, muitos SGBDs permitem que programas ou procedimentos sejam escritos pelo usuário, simplificando o processo de obtenção e classificação das informações. Esses procedimentos são mais que simples conjuntos de comandos padrão ou de consulta: vêm escritos em linguagens de programação completas, embora limitadas, incluídas no SGBD. Algumas dessas linguagens assemelham-se às mais comuns de programação e não são difíceis de aprender.

O dBase II, da Ashton Tate, por exemplo, incorpora uma linguagem fácil de aprender e de empregar. Suas estruturas mostram algumas semelhanças às do BASIC; razoavelmente simples, incluem DO WHILE-ENDDO, DO CASE-ENDCASE e IF-ENDIF. Além disso, existem cerca de cem comandos — alguns familiares, outros desenvolvidos especificamente para uso em bancos de dados. Entre os comandos familiares estão: CALL (passar o controle para uma rotina em linguagem de máquina); NOTE (equivalente ao REM do BASIC); STORE <expressão> TO <variável> (equivalente a LET <variável> = <expressão> em BASIC); e muitos outros. Comandos menos familiares incluem o SKIP (avanço ou retrocesso pelos registros do arquivo) e o FIND <string de caracteres>.

Também são fornecidas várias funções, algumas semelhantes às do BASIC. Por exemplo: CHR <expressão> (que equivale a CHR$(x)); LEN <expressão em string de caracteres> (equivalente a LEN em BASIC); TYPE <expressão> (retorna o tipo da expressão); e DATE() (uma variável do sistema contendo a data).

Para extrair informações, um procedimento feito sob medida é mais eficaz que uma sequência de comandos introduzidos a partir do teclado. Podemos comprovar isso criando um banco de dados de estoque e inventário com o dBase II e escrevendo um procedimento nessa linguagem. Os registros no arquivo serão muito simples, com quatro campos para caracteres numéricos e um para não numéricos:

CÓDIGO DO FABRICANTE	06116
CODIGO DA PEÇA	3995
QUANTIDADE	86
PREÇO	34.750
DESCRIÇÃO	DISCO DE FREiC

Em primeiro lugar, precisamos encontrar uma só palavra para cada campo, estabelecer os valores que queremos para eles e determinar se são de caracteres "não numéricos", "numéricos" ou "lógicos". Isto será suficiente:

CODFABR	: 5 caracteres; numérico
CODPECA	: 5 caracteres; numérico
QUANTIDADE	: 3 caracteres; numérico
PRECO	: 6 caracteres; numérico
DESCRICAO	: 40 caracteres; não numérico

Quando o dBase II começa a rodar, surge um ponto na tela indicando que você deve entrar com os dados. O comando CREATE inicia um novo arquivo, que denominaremos PECAS, por exemplo. O processo será assim:

```
. create PECAS <CR>
ENTRAR COM ESTRUTURA DO REGISTRO
  COMO SEGUE:
CAMPO NOME, TIPO, LARGURA, CASAS
  DECIMAIS
001 codfabr,n,5,0
002 codpeca,n,5,0
003 quantidade,n,3,0
004 preco,n,6,2
005 descricao,c,40
006 <CR>
```

Nossos dados foram introduzidos em minúsculas, e a resposta do dBase II foi impressa em maiúsculas. Repare que o número de casas decimais dos campos numéricos — bem como o ta-

manho do campo — deve ser especificado. Se a tecla [Return] for pressionada sem que se tenha entrado com uma informação — tal como foi feito com o campo 006 —, a fase inicial da criação de arquivos será encerrada.

Utilizamos, a seguir, o comando APPEND. Tem o efeito de limpar a tela e exibir um "esqueleto" do registro, à espera de dados.

```
CODFABR    :         :
CODPECA    :         :
QUANTIDADE :  :
PRECO      :            :
DESCRICAO:
```

Podemos, agora, digitar os dados para cada campo. Um [Return] indica que se completou a entrada num campo; logo que um [Return] for pressionado para o último campo, todos os dados serão aceitos, e o esqueleto do registro reaparecerá, esperando pelo conteúdo do registro seguinte. Para terminar a entrada de dados, aperta-se [Return] no início de um registro vazio.

Introduzidos os dados no arquivo, vamos ver agora como utilizá-lo. Suponhamos que você precise saber em detalhe o valor de varejo de todo o seu estoque, para fins contábeis. Você poderia ver na tela, um a um, todos os registros, e anotar o número de peças ainda em estoque; a seguir, registraria o preço unitário e multiplicaria os dois números. Depois de percorrer todos os registros, o valor total seria obtido pela soma de todos os subtotais. Se você estiver utilizando um fichário convencional, essa poderá ser a única maneira de chegar a uma solução.

A maioria dos SGBDs permite obter esse tipo de informação de modo muito mais simples. Eis como poderia ser feito com o dBase II:

```
.use pecas
.sum quantidade * preco
37.870,58
```

Na primeira linha, um comando instrui o dBase II para trabalhar com o arquivo denominado PECAS. A segunda linha pede para "somar o resultado da multiplicação do campo QUANTIDADE pelo campo PRECO, em cada registro". Na terceira, você vê o tipo de resposta que poderá aparecer após o valor total, em dinheiro, do estoque todo. Melhor que o fichário, não?

Esse exemplo ilustra com clareza como pode ser útil um SGBD eficiente, mesmo utilizando apenas um simples comando. Mas sua capacidade efetiva se manifesta quando armazena sequências de comandos sob a forma de programas. Para armazenar num programa — que chamaremos de VALOR — nossos comandos simples de avaliação de estoque, seria necessário o seguinte diálogo:

```
.modify command
ENTRE NOME DO ARQUIVO: valor
set talk off
use pecas
sum quantidade * preco
set talk on
cancel
```

Esse curto programa será armazenado em disco e poderá ser usado digitando-se o comando DO:

```
.do valor
```

O programa será lido e executado automaticamente. Tão logo seja encerrado, o controle retornará ao dBase II.

Procedimento de consultoria

```
* Mostra consultores
SET TALK OFF
USE SOCIOS
DO WHILE.NOT.EOF
  IF PROFISSAO = "CONSULTOR"
    DISPLAY NOME
  ENDIF
  SKIP
ENDDO
```

Este programa em dBase II extrai os nomes dos consultores participantes de um clube. A linha inicial, com um asterisco, indica que se trata de um comentário. Não precisamos que todos os números de registro sejam exibidos à medida que se efetua a busca. Por esse motivo, comandamos SET TALK OFF.

Devemos especificar que o arquivo procurado é o SOCIOS, pois podem existir numerosos arquivos diferentes no banco de dados. Para que o dBase II procure através dos registros disponíveis, construímos um loop em torno da estrutura DO WHILE-ENDDO.

Dentro do loop, uma condição estabelece que, quando o campo profissão contiver o string "CONSULTOR", o programa exibirá o campo do nome do registro. A condição IF deve ser encerrada por ENDIF. O programa é então entregue a um comando SKIP, que diz ao dBase II para ir ao registro seguinte.

O PROTOCOLO MIDI

Excelente auxiliar nas primeiras experiências com música eletrônica, o seqüenciador analógico foi suplantado pelo protocolo MIDI, que deu aos músicos mais versatilidade e recursos quase ilimitados.

O seqüenciador analógico causou enorme impacto na composição de música eletrônica, nas apresentações ao vivo e nas gravações de discos. Mas ele apresentava a desvantagem de controlar uma gama de variáveis em apenas um sintetizador. Um músico que, por exemplo, possuísse dois sintetizadores — um com poderosa capacidade de seqüenciação e outro com boa qualidade de som — não poderia conectá-los de modo a combinar suas características num único equipamento. O mesmo ocorria nos estúdios de gravação, que não dispunham de um meio de coordenar uma grande variedade de sons gerados por diferentes equipamentos.

A tentativa de solução para esse problema levou à criação de uma interface digital para centralizar o controle de todo o sistema numa única máquina. A MIDI (Musical Instrument Digital Interface, "interface digital para instrumentos musicais") tornou possível, na produção musical, o controle de qualquer sistema digital por outros equipamentos, como, por exemplo, microcomputadores.

Desenvolvido em função de um acordo estabelecido entre os principais fabricantes de instrumentos digitais do Japão e dos Estados Unidos, o atual protocolo, ou especificação, foi concluído em agosto de 1983. Trata-se de um circuito de controle que transmite dados de um instrumento para outro ou de um microcomputador para um instrumento musical.

A MIDI funciona com um processador de 8 bits, e seus projetistas foram obrigados a escolher entre os dois modos de transmissão disponíveis — paralelo ou serial. Na transmissão paralela, é necessária uma linha para cada bit, de forma que os 8 bits de cada byte são enviados simultaneamente, possibilitando elevada velocidade de transmissão. Suas desvantagens estão no alto custo e na inconveniência de se terem ligações de, no mínimo, oito fios ligados a um conector de 25 pinos.

A transmissão serial, por outro lado, emprega apenas duas linhas: numa delas, enviam-se os bits de dados um atrás do outro; a outra serve para que o instrumento receptor acuse erros de paridade no fluxo de dados. Embora mais lenta que a paralela, a transmissão serial utiliza conector mais simples e é bem mais barata.

A motivação inicial para a criação da MIDI foi a necessidade de um circuito de controle que realçasse a música sintetizada e proporcionasse algum tipo de comando de tonalidade. Além disso, o preço deveria ser acessível aos proprietários de microcomputadores e músicos que usam sintetizadores e baterias eletrônicas. Por estas razões, os fabricantes decidiram pelo uso da transmissão serial na MIDI.

Essa interface transmite em modo assíncrono, como a RS232C (padrão de muitos microcomputadores para conexão a modems ou impressoras seriais). A transmissão realiza-se por intermédio de um chip Motorola 6850 ACIA (Asynchronous Communications Interface Adaptor, "adaptador para interface de comunicações assíncronas"), que acrescenta 2 bits a cada byte do instrumento dominante. O primeiro é "0" (bit de partida), seguido pelos 8 bits do dado serial, terminando tudo com "1", o bit de parada. Essa palavra serial de 10 bits é então enviada ao instrumento receptor, onde um segundo chip 6850 a converte aos 8 bits originais.

Ligação perfeita

Glissando
Para conseguir o glissando num instrumento de cordas, basta deslizar os dedos sobre elas até a nota seguinte. Mas, para produzir o mesmo efeito sonoro num sintetizador, é preciso uma programação bastante complexa.

Por ser caro, esse chip fica protegido no circuito por opto-isoladores (dispositivos dotados de células fotoelétricas que permitem a troca de sinais entre dois circuitos eletricamente isolados); assim, os picos de tensão não provocam qualquer dano ao microprocessador.

A MIDI difere da interface RS232 num ponto importante: a velocidade de transmissão da RS232 chega a 19,2 Kbauds, enquanto a da interface para instrumentos musicais é somente a metade. Isso não afeta a compatibilidade com os microcomputadores, pois os circuitos lógicos internos da MIDI ajustam a velocidade de sincronização para 31,25 Kbauds (relativamente alta para transmissões seriais).

Como a interface MIDI coordena mais de um instrumento, sua primeira tarefa é enviar o dado certo ao instrumento apropriado — caso contrário, uma bateria eletrônica acabará tocando uma melodia cuidadosamente seqüenciada, enquanto um sintetizador polifônico tentará reproduzir uma batida sincopada de bateria.

Em função disso, os instrumentos compatíveis com a MIDI dispõem de um código de identificação numérico. Um dos dezesseis canais disponíveis na interface destina-se ao código, de forma que somente esse canal aceita dados para determinado instrumento. Assim, a primeira parte de uma transmissão completa é um byte de status que inclui o identificador. Todos os dados que seguem essa instrução de encaminhamento especificam, portanto, como o comando deve ser interpretado pelo instrumento.

A interface possui dois conectores DIN de cinco pinos, denominados MIDI IN e MIDI OUT. O primeiro recebe instruções de um microcomputador ou sintetizador mestre e o outro transmite a seqüência modificada de bits para o instrumento receptor.

Muitos modelos contam ainda com o conector MIDI THRU, uma saída que transmite sem alterações o fluxo de bits (enviado para o MIDI IN). Esse fluxo pode então ser dirigido para uma segunda interface. O cabo, com 15 cm de comprimento máximo e conectores DIN de cinco pinos, possibilita a ligação a um microcomputador ou ao sintetizador mestre.

Num sistema musical controlado por computador, este pode ter dois papéis que dependem do tipo de MIDI utilizado. Na primeira alternativa, o micro apenas fornece memória para a gravação da música e ativa a interface. Na outra, ele contém o software da MIDI. Essa interface pode ter dois tipos de atuação: conecta os instrumentos ao micro ou age sobre o fluxo de dados. Esse fluxo existe sob duas formas: de gravação e de reprodução. No primeiro caso, a música é transmitida à memória pela MIDI; no segundo, antes que ela chegue aos instrumentos, a interface produz modificações de ritmo, timbre etc., conforme as informações de controle definidas pelo usuário.

O dó central

Imagine alguém que usa pela primeira vez a MIDI e pretende tocar uma melodia curta. Esta começa com um dó central, sobe para mi, depois para sol e assim por diante. O modo de programar depende do tipo de software musical em uso. As notas são introduzidas com uma caneta óptica na pauta de cinco linhas apresentada no vídeo. Ou, então, por meio do teclado do micro, segundo algum tipo de MCL (Musical Composition Language, "linguagem de composição musical"). Uma alternativa seria tocar as notas num teclado musical, ligado ao micro, que, sem gerar som algum, produzisse alguma das representações resumidas acima.

Introduzida a música, a transmissão da MIDI será sempre a mesma: para iniciar a melodia, o primeiro byte — transmitido como mensagem serial com seus 2 bits extras a partir do chip ACIA — seleciona, por exemplo, o canal 6; o segundo byte, o dó central.

Essa instrução produzirá a nota dó central no sintetizador, que a reproduzirá sem parar, a menos que haja outra instrução para interromper ou limitar sua duração. Não havendo, e se o resto

Formato apropriado
O chip 2764 contém e aplica o protocolo MIDI aos sinais que entram e saem da interface.

Entrada/saída
O processador HD6801 controla essas funções básicas.

Processador dual
O chip 6116 tem um buffer bidirecional e um relé de seqüenciamento digital. Ele também controla as interrupções.

Interface interna
O chip HD61 controla as portas da MIDI, os sinais de entrada e o endereçamento de ROM e RAM.

MIDI
Essa interface identifica e coordena os protocolos de entrada/saída de um computador e dos instrumentos musicais ligados a ele. Também processa o som digitalizado, incluindo, nos dados que o definem, informações de controle, sincronização e ritmo para serem transmitidas a um sintetizador.

da melodia for introduzido sem parâmetros de duração, todas as notas continuarão a soar ao mesmo tempo. O resultado será um acorde ininterrupto, composto de todas as notas da melodia.

Um mínimo de familiaridade com o pentagrama na tela basta para detectar os eventuais enganos que forem cometidos. Assim, para corrigi-los, um usuário de MCL executaria a seqüência desejada enviando uma instrução monofônica ao sintetizador.

"Monofonia" quer dizer apenas um som — em contraposição a "polifonia" (muitos sons). Quando o sintetizador receptor produz um som por vez (opção pelo modo monofônico), a seqüência será executada como uma sucessão de notas isoladas — uma melodia, em vez de um acorde.

Composição eletrônica

Suponha que, após várias tentativas e erros, um principiante tenha digitado corretamente sua música e que o sintetizador a reproduza. A melodia mantém o ritmo — definido pela seqüência de durações — e o compasso corretos. Observe que, até agora, neste artigo, os tipos de instrução têm sido muito limitados. Apenas duas características (ou parâmetros musicais) foram mencionadas — freqüência (dó central, mi, sol etc.) e duração.

O compositor da melodia ouve-a algumas vezes e acha que ela soa muito "dura" — como é provável ocorrer enquanto se tem uma definição mínima. Decide que, ao invés de os primeiros dó e mi ocorrerem um após o outro, o dó deve durar até o início do mi. Esse movimento, conhecido como "glissando" caracteriza a forma pela qual se alonga uma nota. Desse modo, o usuário acrescenta um toque de vivacidade ao trabalho do sintetizador.

Isso conduz a um novo problema: se o sintetizador receptor não possui recursos para realizar glissandos, não executará a instrução. Poderá tocar o dó central como se estivesse recebendo a instrução original, ou fazer algo totalmente diferente.

Se o usuário da MIDI der instruções para a produção de um trecho de música polifônica e o sintetizador for monofônico, este fará uma imprevisível seleção da polifonia e a executará monofonicamente. Em resumo, o uso da MIDI para acoplar um micro a um sintetizador extremamente elementar não irá transformá-lo num equipamento sofisticado como o Fairlight.

Essas restrições valem também para o processo inverso: o sintetizador receptor pode ter excelente qualidade, mas, a não ser que se definam parâmetros musicais suficientes — e a menos que se ajustem corretamente os próprios controles do sintetizador —, o resultado será equivalente à musicalidade de uma calculadora de bolso.

Na prática, é possível amenizar essa incompatibilidade: por meio dos controles do sintetizador, estabelecemos como constantes o maior número possível de parâmetros, para que as instruções da MIDI se mantenham de acordo com eles. Essa abordagem tem maior probabilidade de ser adotada pelo usuário do sintetizador.

Para enriquecer a execução, a interface tem, em tese, 128 controles — filtragens, distorção, "ruído branco" (todas as freqüências possíveis) e "ruído rosa" (freqüências médias), cada um com valores entre 0 e 128 —, suficientes para abranger os parâmetros disponíveis na maioria dos sintetizadores.

Questão de velocidade

A velocidade de transmissão se torna um problema. Um comando relativo a uma única nota, definindo-se somente dois parâmetros, usa três palavras seriais. Leva quase 1 milissegundo para ser executado à razão de 31,25 Kbauds. Como os acordes de seis notas são comuns em muitos gêneros musicais, um acorde desses exige 5,76 milissegundos para ser transmitido.

Quando se define esse acorde com mais precisão, por meio dos demais controles da MIDI, o tempo torna-se longo o suficiente para que o ouvido humano comece a perceber alterações nas características do som. Essas alterações ficam evidentes apenas quando os sons (principalmente os semelhantes) estão próximos — e a interface MIDI foi projetada para manipular sons simultâneos.

A música, no entanto, é um meio de comunicação "paralelo": as pessoas acostumaram-se a ouvir sons simultâneos. Portanto, não surpreende que a transmissão serial da interface tenha sido criticada — a paralela certamente seria muito mais adequada. No momento, porém, essa característica se deve a um compromisso entre custo e eficiência. Afinal a MIDI, com suas especificações atuais, está apenas em sua primeira versão.

OSBORNE ENCORE

Fabricante do primeiro micro portátil, a Osborne recuperou-se de seus problemas financeiros e lançou o Encore — mais um integrante da extensa linha de máquinas compatíveis com o IBM PC.

Primeiro computador portátil integrado, o Osborne-1 iniciou uma revolução na microcomputação. Equipado com monitor incorporado, duas unidades de disco e interfaces para modems e impressoras, foi também o primeiro computador profissional autônomo operando com CP/M. Embora sua classificação como "portátil" fosse um tanto exagerada (o Osborne-1 pesava 10,5 kg), outros fabricantes logo perceberam o potencial da nova máquina. Em pouco tempo, a Osborne estava rodeada de concorrentes.

O aviso de novos tempos veio em 1981, quando a IBM anunciou o lançamento de seu primeiro modelo na linha de computadores pessoais. A entrada da gigante americana na produção de micros arrebatou enorme fatia do mercado, atraída pela fama da empresa. Se, por um lado, os micros da IBM provocaram o desaparecimento de algumas marcas, por outro, representaram enorme avanço na indústria de computadores, estimulando outros fabricantes a produzirem equipamentos compatíveis com o IBM. Tal foi o caso da Osborne, que, juntamente com outros competidores, não perdeu tempo e anunciou um novo modelo, o Osborne Executive, compatível com o IBM PC.

Esse equipamento vinha com dois processadores, permitindo executar programas aplicativos para CP/M e para MS-DOS. No entanto, devido à escassez mundial do microprocessador 8086, a máquina foi lançada sem o chip necessário para torná-la compatível com o IBM PC. Como resultado, as vendas caíram vertiginosamente, quase obrigando a empresa a fechar suas portas em meados de 1983. Mas a Osborne conseguiu sobreviver, dedicando-se à pesquisa e ao desenvolvimento.

Para uma empresa que quase chegou a falir e mal conseguiu se manter à tona no competitivo mercado de computadores, seu novo lançamento — o Osborne Encore — revela-se uma audaciosa jogada. Mesmo não sendo tão compacto quanto outras máquinas como, por exemplo, o Epson PX-8, o Encore destaca-se por situar-se entre os dois segmentos do mercado: os equipamentos de mesa e os pequenos portáteis.

O Encore pesa 6 kg, bem menos que seu predecessor. O teclado se dobra contra a tela, formando uma caixa compacta, do tamanho aproximado de três listas telefônicas.

O teclado é do tipo máquina de escrever; acima dele há um painel tipo membrana, sensível ao toque, com as teclas de função e os "ícones" (símbolos que representam os programas residentes). O corpo do computador abriga também um visor de cristal líquido de 23,7 x 8 cm. As teclas de controle encontram-se nos dois lados, enquanto as quatro teclas do cursor dispõem-se no canto inferior direito.

Tem-se a impressão de que as teclas se comprimem numa área muito estreita. Isso resulta da tentativa de colocar, num espaço reduzido, todas as características do IBM PC, que tem, à direita, um teclado numérico separado. Esse conjunto, com vistas a manter a compatibilidade, foi incorporado à parte principal do teclado do Encore. As teclas numéricas estão marcadas em azul, contrastando com o branco das alfanuméricas. Para utilizar as numéricas e os outros programas residentes, tocam-se antes os ícones correspondentes. Estes, no Encore, representam as rotinas de cálculo, modem, disco e o programa calendário.

O projeto do painel de toque não foi muito feliz, depreciando o ar profissional do teclado.

Compacto
O Encore da Osborne é uma das primeiras máquinas compatíveis com o IBM PC equipadas com visor de cristal líquido em substituição ao monitor de vídeo convencional. Quando não está em uso, o teclado se fecha contra a tela, formando um volume bastante compacto. Pode-se carregá-lo a tiracolo pela alça.

As teclas de função (que no IBM PC estão separadas à esquerda) são do tipo membrana, desagradáveis ao tato. Embora se possa sentir o movimento de membrana sob o painel, as teclas carecem do toque firme de um teclado profissional.

Incorporando um visor de cristal líquido (LCD) os projetistas obtiveram grande economia no consumo de energia, visto que gasta bem menos que o tubo de raios catódicos usual. Aí residem os maiores problemas de compatibilidade com o IBM PC, não por deficiência nas cores resultantes do visor de cristal líquido, pois estas podem ser facilmente substituídas variando-se o sombreado dos gráficos; a desvantagem está no pequeno tamanho da tela.

A tela normal do IBM PC tem, em modo texto, uma resolução de 80 colunas x 25 linhas; mas, devido aos problemas encontrados pelos fabricantes japoneses no desenvolvimento de um LCD equivalente, a Osborne viu-se forçada a equipar o Encore com um visor de 80 colunas x 16 linhas. Em conseqüência, muitos programas aplicativos escritos para o IBM PC não podem ser utilizados no seu equipamento.

Isso não constitui problema para os programas que rolam verticalmente na tela; mas, em pacotes onde o visor é "paginado", o usuário, em geral, encontrará sérias dificuldades, pois as mensagens aparecem na parte inferior do vídeo. Comparado ao monitor de vídeo, o LCD ainda apresenta outras desvantagens, como a necessidade de forte iluminação para produzir caracteres legíveis e o tempo de familiarização que exige do usuário.

Do lado direito do computador, há espaço para duas unidades de disco flexível de 5 1/4 polegadas; contudo, o modelo básico vem equipado com apenas uma unidade. Do lado oposto situam-se o encaixe para o transformador de energia, uma chave liga/desliga, um botão para ajuste do contraste da tela, e o alojamento das baterias — fontes de energia especiais, de cádmio-níquel, que permitem o funcionamento da máquina por um período de 4 a 5 horas.

Conexões

Na parte posterior do Encore estão os conectores: um telefônico, que se liga ao modem interno, uma saída paralela padrão Centronics para impressora e uma saída serial padrão RS232 para conexão com outros dispositivos, como impressoras seriais e modems externos. Para carregar o disco de sistema operacional MS-DOS, basta pressionar no painel de toque o ícone que representa um disco.

Todos os programas em disco do IBM PC podem ser lidos pelo Encore. No entanto, devido às restrições da tela, é difícil detectar se a execução está correta. Entre os programas que o Encore executa de forma bem-sucedida está o Lotus 1-2-3 — difícil de se rodar num compatível, por sua maneira pouco ortodoxa de acessar as rotinas residentes no IBM.

Alto-falante
Recursos visuais são enfatizados por sinais sonoros

Visor de cristal líquido
O LCD exige muito menos energia que os monitores de vídeo convencionais, tornando realidade o compatível IBM PC acionado a bateria.

Funções duplas
O teclado do Encore contém poucas teclas a menos que o do IBM PC. Mas fornece as mesmas funções agrupando duas delas na mesma tecla

Compacto, mas incômodo

As unidades de disco (podem ser usadas até duas) situam-se na lateral da máquina, tornando-a muito compacta. Para o usuário, no entanto, isso constitui um aborrecimento, pois obriga-o a esticar-se para inserir um disco ou verificar se a unidade certa está sendo acessada.

Alimentação
Um transformador externo permite usar energia da rede elétrica.

Placa principal
Observe-se o acondicionamento compacto dos chips, que permitiu acomodar os 512 Kbytes de memória e outros recursos num reduzido espaço.

Unidades de disco
O Encore pode ser equipado com uma ou duas unidades de disco. O invólucro metálico funciona como dissipador de calor.

Saída para impressora
É a saída paralela padrão Centronics

Saída RS232
Conecta o Encore a impressoras seriais ou modems

OSBORNE ENCORE

MICROPROCESSADOR

8086 de 16 bits

MEMÓRIA

RAM de 521 Kbytes.

SISTEMA OPERACIONAL

MS-DOS.

VISOR

Visor de cristal líquido. Modo texto: 80 x 16 caracteres; modo gráfico: 480 x 128 pixels.

TECLADO

Com 62 teclas tipo máquina de escrever, mais um painel de toque tipo membrana com dez teclas de função e quatro de ícones.

LINGUAGENS

BASIC da Microsoft, fornecido em disco.

INTERFACES

Saída serial RS232, saída paralela padrão Centronics e conector telefônico RJ11.

DOCUMENTAÇÃO

Os dois guias do usuário são, como a maioria dos manuais da Osborne, muito bem escritos, com amplas explicações sobre o funcionamento da máquina e vários exemplos.

Controlador de tempo
Na foto, o visor de cristal líquido de 80 x 16. O espaço delimitado na parte inferior destina-se a acomodar a futura tela de 80 x 25. O visor com calendário/agenda permite registrar compromissos futuros nos diferentes horários das regiões do mapa.

Sensível ao toque
Acima do teclado principal está o painel de toque, tipo membrana — semelhante ao da linha Sinclair ÷, contendo dez teclas de funções programáveis. Abaixo destas estão os ícones, usados para acionar os programas gravados na ROM do Encore.

RELAÇÕES PRODUTIVAS

Os bancos de dados implementados em microcomputadores baseiam-se, em geral, nos métodos de organização "relacional". Existem métodos mais eficazes, embora mais complexos: os sistemas hierárquico e em rede.

A maioria dos SGBDs (Sistemas Gerenciadores de Bancos de Dados) disponíveis para microcomputadores pertence ao tipo conhecido como sistema "relacional". Em nível bem simples, isso significa que cada registro num arquivo SGBD assume a forma de uma tabela, com linhas e colunas, semelhante a uma planilha financeira. As diversas maneiras de apresentar as informações podem sugerir estruturas mais complexas, mas, em última análise, estas não passam de tabelas. A organização do banco de dados assemelha-se, em geral, ao modo pelo qual você distribuiria as informações numa folha de papel.

Todos os registros possuem a mesma estrutura e cada campo no interior dos registros ocupa um espaço prefixado. O termo "relacional" provém do fato de estarem as "linhas", no banco de dados, relacionadas às "colunas". Essa não constitui uma forma muito flexível de organizar dados, mas dela resultam programas relativamente simples para manipulação de bancos de dados. As limitações de um sistema relacional fazem parte do preço a ser pago quando se combinam SGBDs e recursos não muito amplos de processamento e capacidade de memória, como os oferecidos pela maioria dos micros — mesmo as novas máquinas de 16 bits.

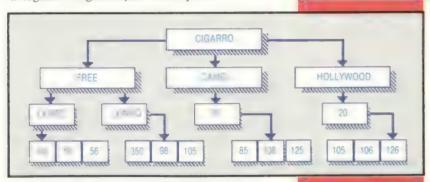
Uma abordagem diferente consistiria em ordenar os elementos em forma hierárquica. Imagine os dados organizados numa estrutura semelhante a uma árvore, cujos galhos dessem origem a subgalhos, os quais, por sua vez, se dividiriam em ramos e, finalmente, folhas.

Para ilustrar esses dois tipos de organização de dados, vamos apresentar — primeiro de modo relacional, depois de maneira hierárquica — o estoque de um empório:

Gráfico em árvore
Enquanto um SGBD "relacional" armazena dados sob forma tabular, um "hierárquico" organiza as informações em forma ramificada. Na figura, a categoria "remédio" contém indicadores para os vários tipos de remédio disponíveis, cada um com indicadores para diversas marcas comerciais, e assim por diante. Os SGBDs hierárquicos, embora muito eficientes em seu uso de memória, são mais complicados em termos de estrutura do que os equivalentes relacionais.

CATEGORIA	MARCA	QUANTIDADE/ UNIDADE	ESTOQUE	CUSTO	VAREJO
CIGARRO	GALAXY	20	350	98	105
CIGARRO	FREE	20	140	50	56
CIGARRO	CAMEL	20	85	108	125
CIGARRO	HOLLYWOOD	20	105	106	126
CHOCOLATE	LACTA	1	106	14	17
CHOCOLATE	GALAK	1	95	12	16
CHOCOLATE	PRESTÍGIO	1	25	15	19
REVISTA	CRIATIVA	1	35	85	95
REVISTA	ELE&ELA	1	12	60	80
REVISTA	MODA	1	86	50	60
BEBIDA	OLD EIGHT	1.000 ml	35	25	37
BEBIDA	SMIRNOFF	500 ml	40	18	27
BEBIDA	COCA-COLA	180 ml	20	16	25

Na representação hierárquica, seria assim o banco de dados para o estoque de produtos:

Ao invés de ser organizado em campos, o registro divide-se em "segmentos" de dados, cada um dos quais contendo mais de um campo. Os itens introduzidos que correspondem a CIGARRO no banco de dados do tipo relacional apresentariam a seguinte organização hierárquica:

Nesse tipo de representação, é possível reconhecer cada item de informação como superior ou subordinado a outro item. Para isso, precisa-se apenas de indicadores que "apontem" de um segmento de dados para outro. Neste exemplo, precisamos apenas de um segmento CIGARRO, com indicadores para os vários tipos de cigarro em estoque, e indicadores adicionais para o número de cigarros por maço (quantidade por unidade), custos, preços e assim por diante.

Vamos supor que o empório tenha apenas cem "categorias" para a classificação das mercadorias, mas um catálogo de 1.000 artigos à venda; com um banco de dados relacional, seriam necessários 1.000 registros — um para cada artigo. Com um banco de dados hierárquico, porém, apenas cem segmentos abrangem todo o estoque. Isso evita um grande número de duplicações, mas torna o banco de dados estruturalmente muito mais complicado.

Existe outro tipo de organização de banco de dados, um sistema em rede que foi padronizado na CODASYL (Conference on Data Systems Languages, "Conferência sobre Linguagens de Sistemas de Dados"), ficando conhecido com esse nome. O problema dos bancos de dados hierárquicos é que, neles, os dados fluem num único "sentido" (um ramo não pode estar ligado a dois troncos). Na maioria dos casos, isso não acarreta confusões, embora possam existir cigarros com nomes idênticos aos de fábricas de discos etc. As dificuldades aparecem quando uma peça tem mais de um fabricante. Se você vende peças de automóveis, um dispositivo elétrico para um Volks pode ser fabricado pela Peçasmil, Vidrofumê, ou pela Volksmar.

No sistema CODASYL, utiliza-se uma rede de conjuntos, cada um consistindo numa coleção de registros. Ao contrário do sistema relacional, o tamanho do registro não precisa ser fixo. Um conjunto pode incluir apenas um registro, e qualquer registro pode pertencer a mais de um conjunto — desde que não pertença a mais de uma ocorrência do mesmo tipo de conjunto. Assim, não são permitidas estruturas como esta:

Essa é uma limitação séria. Embora sejam mínimas as chances de haver CIGARROS, REVISTAS e BEBIDAS com nomes idênticos, é fácil imaginar uma companhia chamada Supertrigo Ltda., que faça parte de uma estrutura semelhante a esta:

Embora mais flexíveis que os relacionais, os bancos de dados hierárquicos e os do tipo CODASYL são bem mais complexos e requerem, pelo menos, minicomputadores. Os micros utilizam, quase sempre, sistemas relacionais. A dificuldade mais comum decorrente dessa limitação surge quando, por exemplo, se tem um inventário de peças e cada uma tem mais de um fornecedor. Nesse caso, são necessários registros para as peças e um conjunto de registros de fornecedores. Já os SGBDs de multiarquivos permitem que dois ou mais arquivos sejam abertos simultaneamente, e que um arquivo ativo faça referência a um subsidiário.

TENTATIVA E ERRO

Com uma inspeção mais detalhada de alguns de seus recursos exclusivos, conclui-se aqui o exame do programa de processamento de equações TK!Solver, um sucesso no mercado internacional.

O TK!Solver é um software de novíssima geração, que leva o conceito da planilha eletrônica aos domínios da matemática superior e da engenharia. Permite que o usuário defina variáveis com nomes e use esses nomes em equações matemáticas complexas.

Uma das características marcantes do TK!Solver é sua extraordinária capacidade de iterar, ou seja, de chegar a um resultado por ciclos de repetições.

A iteração é um método no qual o programa determina uma variável por meio de tentativas. Em qualquer situação de cálculo, a pessoa que lida com equações poderá determinar os valores de todas as variáveis se tiver, de início, informações suficientes. O programa simplesmente reduz o problema a uma série de cálculos. Por exemplo, a expressão

$$A^2 + B^2 = 2 \cos Y$$

poderá ser resolvida para qualquer das três variáveis se os outros dois valores forem conhecidos. Em face dessa equação — e dados os valores de A e B —, o TK!Solver executa os cálculos necessários e encontra o valor de Y.

Mas há ocasiões em que a determinação do valor não é imediata. Isso ocorre, por exemplo, na equação redundante, que define uma variável em termos dela mesma. Consideremos, por exemplo,

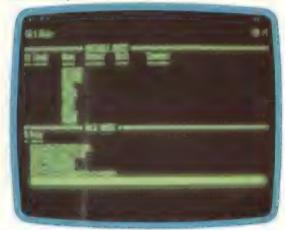
$$D = (A + B)/(2 * D)$$

num modelo em que A é o único valor conhecido. Podem ocorrer outros problemas, como resultado de um modelo incompleto ou de um modelo com muitas variáveis interdependentes e uma quantidade limitada de dados.

O modelo da viagem de automóvel, citado em matéria anterior desta série, envolve cinco valores: distância, tempo, velocidade, combustível e autonomia (milhagem). O programa calculava esta última, dados a velocidade e o consumo de combustível; a distância, a partir da velocidade e do tempo; e outras variações simples. Como proceder se quisermos calcular a velocidade que deve ser mantida a fim de completar a viagem dentro de um orçamento predeterminado?

Para começar, acrescentam-se alguns fatores ao modelo. Por exemplo, o modelo tem de considerar a potência do veículo, o atrito interno do motor e a resistência do ar. (Suporemos o atrito interno constante.) Deve-se também estabelecer um teto para o orçamento e o custo do combustível a consumir.

Começaremos a montar o modelo introduzindo essas equações na planilha Regras, uma por linha. As equações são automaticamente transferidas para a planilha Variáveis *(tela 1)*.

Tela 1: Montagem de equações

O tamanho da tela é insuficiente para mostrar as muitas variáveis. Podem-se apresentar todas as informações, inserindo a planilha Variáveis numa janela própria. Para tanto, pressionamos a tecla [;], colocando o cursor na área das variáveis, e digitamos W1 (window, janela). Agora, todas as variáveis estão visíveis e podemos atribuir-lhes valores.

Solução direta

O modelo será solucionado diretamente se, de início, conseguir informações suficientes. Por exemplo, introduzem-se na coluna Input (entra-

Tela 2: Valores de entrada no Direct Solver

da) valores relacionados aos nomes distância, velocidade, preço, atrito e vento (resistência do ar) *(tela 2)*. Pressiona-se, então, [!] para efetuar os

Tela 3: Resultados do Direct Solver

cálculos. O TK!Solver apresenta a mensagem Direct Solver, e os valores das incógnitas surgirão na coluna Output (saída), como se vê na *tela 3*.

Isso dá ao usuário um bom detalhamento das informações desejadas. Mas o que ele fará se quiser usar esse modelo para resolver um problema com menos informações?

Consideremos um cálculo que envolva um máximo de 50 libras para gastar em combustível numa viagem de 1.000 milhas. Sabe-se que o galão de combustível custa 1,75 libra. Assim, determinamos quanto se pode gastar por milha. Começamos apagando os valores já introduzidos, pela digitação de RVY (Reset Variables Yes, "reinicializar variáveis"). Introduzimos as informações conhecidas: 1.000 para milhagem, 50 para o custo e 1,75 para o preço. Usamos o valor 1/3 como atrito interno (que o TK!Solver transforma em 0,333333) e 0,000095 para a resistência do ar. Pressionando [!] para que o programa execute

Tela 4: Modelo incompleto

o cálculo, surgem então valores para milhagem e combustível *(tela 4)*.

Observa-se que não foram gerados valores para velocidade, tempo e potência — e a velocidade é o valor particularmente necessário. Se o usuário passar da planilha Variáveis para a pla-

Tela 5: Equações não satisfeitas

nilha Regras, verá que, de cinco equações, três permanecem não satisfeitas, como indicado *(tela 5)* por um asterisco na coluna S (status).

Resolução iterativa

Já que não se pode resolver o modelo pela solução direta (Direct Solver), deve-se tentar a iterativa (Iterative Solver). Esta toma um valor inicial e o introduz como tentativa na equação. Se o valor não for correto, o TK!Solver usará uma série de aproximações sucessivas (como no jogo maior-ou-menor), até localizar o valor exato.

A primeira providência é tomar o valor de milhagem gerado anteriormente e colocá-lo na coluna Input. Fazemos isso digitando [I] na coluna St antes da milhagem, na planilha Variáveis. Então estimamos um valor para a velocidade — 50, por exemplo — e o introduzimos na coluna Input. Digitamos G (guess, tentativa) na coluna St e [!] para que se efetuem os cálculos. O TK!Sol-

Tela 6: Valores usados nas iterações

ver apresenta as palavras Iterative Solver na parte superior da tela, e conta cada aproximação. Na quarta tentativa, nesse caso, o programa chega aos valores corretos de velocidade, tempo e potência — indicados na *tela 6*.

De acordo com o TK!Solver, é necessária uma velocidade média de pouco mais de 47 milhas por hora para se completar a viagem dentro do orçamento.

GLOSSÁRIO DAS TELAS

MILEAGE = milhagem
DISTANCE = distância
FUEL = combustível
SPEED = velocidade
TIME = tempo
COST = custo
PRICE = preço
POWER = potência
FRICTION = atrito
WIND = vento (resistência do ar)
VARIABLE SHEET = planilha Variáveis
RULE SHEET = planilha Regras

INTRODUÇÃO AO ASSEMBLY

A programação em código de máquina é a chave para se obter o máximo rendimento do microprocessador. Esta primeira parte de nosso curso levará a uma ampla compreensão dos fundamentos da microprogramação.

O código de máquina é uma linguagem de programação e tem este aspecto:

```
INSTK: SBC $D9FA,X:Outport flag value
```

ou:

```
DE23 FD FA D9
```

ou, ainda:

```
11011110 00100011 11111101 11111010 11011001
```

Outras vezes, pode ser encontrado assim:

```
1240 LET ACC=ACC—FLAG (X)
```

e também:

```
PERFORM FLAG-ADJUST THROUGH LOOP 1
```

Todas essas expressões não passam de diferentes versões da mesma instrução em código e, como ele será decifrado por uma máquina de computação, recebe em todos os casos o nome de "código de máquina". Na realidade, para a máquina, ele se apresenta simplesmente como uma configuração de níveis de voltagem ou de corrente elétrica.

Quando nos referimos ao código de máquina, em geral pensamos na linguagem ASSEMBLY, cujo exemplo é a primeira instrução da lista — para o microprocessador 6502; as outras constituem diferentes formas de representar a mesma seqüência de eventos elétricos.

Portanto, o código de máquina (ou linguagem ASSEMBLY — não vamos nos preocupar com a distinção, por enquanto) é apenas mais uma linguagem de programação. Uma vez que a programação sempre vem antes da linguagem — tanto nos programas em ASSEMBLY do IBM, em BASIC do Atari, como numa hipotética "linguagem telepática venusiana" —, você terá de resolver o problema da programação em sua cabeça antes de mexer no teclado. E o modo como expressará sua solução — o algoritmo — determinará a forma final do programa.

Na verdade, trata-se sempre de escolher, entre várias linguagens, aquela que facilitará o desenvolvimento ou a leitura do programa. A solução, contudo, deve vir antes: o conteúdo precede a forma.

Nesse caso, por que chamá-la de código de máquina e por que se preocupar em usá-la? Damos esse nome à linguagem porque seu conjunto de instruções corresponde com exatidão ao conjunto de operações fundamentais, ou "primitivas", que um determinado microprocessador executa. Por meio do código de máquina, controlamos passo a passo a operação do microprocessador, em vez de deixar que um programa interpretador o faça de modo mais abrangente.

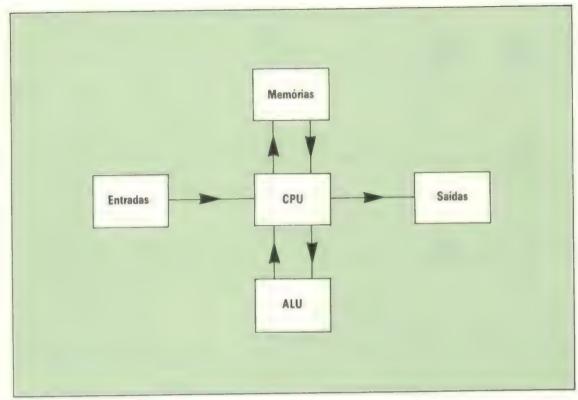
O motivo mais comum para o emprego do código de máquina é a velocidade: ao endereçar diretamente o processador, o programa não precisa ser traduzido ou interpretado. Com a eliminação desse estágio intermediário, reduz-se bastante o tempo de execução do programa. No entanto, o processo de codificação, teste, depuração, modificação e manutenção de um código de máquina exigirá, com certeza, pelo menos duas vezes mais tempo do que levaria num programa em linguagem de alto nível.

A falta de interação com o programador e a dificuldade de se lidar com o código de máquina foi o principal estímulo para a criação das linguagens de alto nível, como COBOL e BASIC. Vimos que o conjunto de instruções em código de máquina equivale ao conjunto de operações do processador. Mas o que são essas operações e o que faz o processador?

O centro do sistema

Em termos simples, a CPU (Central Processing Unit, "unidade central de processamento") do computador não passa de um interruptor que controla o fluxo de corrente num sistema de computação. Compõem esse sistema a memória, a ALU (Arithmetic and Logic Unit, "unidade aritmética e lógica") e os dispositivos de entrada e saída.

Ao acionar uma tecla, você introduz alguma informação na máquina por meio de uma configuração de voltagem gerada na unidade de teclado. A CPU transfere essa configuração de voltagem para uma posição da memória. Em seguida, transfere uma configuração correspondente, proveniente de algum outro lugar da memória, para a tela, de modo a gerar um determinado padrão de caracteres.

Esse processo é semelhante ao funcionamento de uma máquina de escrever, mas com a diferença de que nesta há uma conexão mecânica entre o acionamento de uma tecla e a impressão do caractere, enquanto num computador essa ligação ocorre porque a CPU transfere configurações corretas de voltagem de um lugar para outro.

Controle global
A CPU determina o fluxo de correntes elétricas — e, portanto, de informações · pelos diversos componentes do sistema. Isso permite a realização das quatro operações (aritméticas, lógicas, de memória e de controle) nas quais se baseia o funcionamento de qualquer computador.

Nem sempre o acionamento de uma tecla faz aparecer um caractere no vídeo: pode também destruir um asteróide, gravar um programa, apagar um arquivo em disco, ou imprimir uma carta. O resultado da operação depende do modo e da finalidade com que a CPU transfere a corrente elétrica.

Modo de funcionamento da CPU

Os procedimentos executados pela CPU classificam-se, para nossos objetivos, nas operações: aritméticas, lógicas, de memória e de controle. Todas resultam de transferências de informações através de diferentes trajetos, no sistema e na CPU, ou seja, para esta todas as operações se assemelham.

Operações aritméticas. A adição e a subtração constituem a característica mais importante da máquina. Ela subtrai por meio da representação negativa de um dos números e sua subseqüente adição com o outro. Por exemplo, 7+5=12 significa:

(mais 7) somado a (mais 5) igual a (mais 12);

enquanto 7−5=2 equivale a:

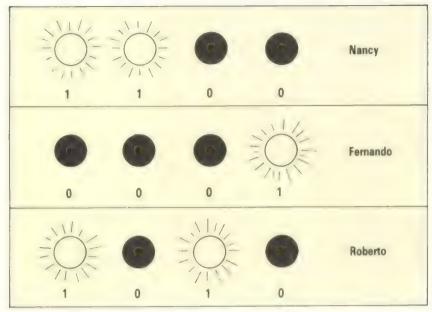
(mais 7) somado a (menos 5) igual a (mais 2)

A multiplicação e a divisão são consideradas adições ou subtrações repetidas, de modo que também é possível programar a CPU para simular essas operações. Se a CPU consegue realizar as quatro opérações aritméticas, então pode efetuar qualquer outro cálculo matemático. No entanto, lembre-se de que todo seu potencial matemático depende simplesmente da capacidade de somar dois números.

Operações lógicas. Efetuam a comparação de dois números: não apenas em termos de quantidades relativas, mas também em termos da configuração de seus dígitos. É fácil ver que sete é maior que cinco porque extraímos cinco de sete e obtemos um resultado positivo. Além de fazer esse tipo de comparação, a CPU também verifica que, por exemplo, 189 e 102 têm o mesmo dígito na coluna das centenas. Talvez isso não pareça algo extraordinário, mas sua utilidade se tornará mais evidente no decorrer do curso.

Operações de memória. Envolvem tanto a cópia de informações de uma posição qualquer da memória externa para sua própria memória (registro), como de seu registro para uma outra posição da RAM. Executando essas operações em seqüência, a CPU transfere informações de uma parte qualquer da memória para outra. Para que a memória do computador tenha alguma utilidade, é absolutamente necessário que a CPU seja capaz de realizar essas duas operações. Só assim torna-se possível um controle completo da memória.

Operações de controle. Consistem, na verdade, em decisões quanto à seqüência pela qual a CPU executa as outras operações descritas. Por enquanto, tudo o que precisamos saber sobre as operações de controle é que a CPU pode tomar determinadas decisões a respeito de sua própria atividade.

Portanto, a CPU executa operações aritméticas, compara números, desloca informações na memória e decide sobre sua própria seqüência de operações. Essa lista de procedimentos é suficiente para definir uma máquina de computação ideal.

Memória elétrica
Um circuito elétrico com interruptores e lâmpadas serve de analogia ao funcionamento da memória nos computadores. Neste exemplo, o gerente de uma fábrica controla a chegada dos funcionários sem sair de seu escritório, apenas associando cada seqüência de luzes acesas e apagadas aos nomes dos empregados.

Se a CPU consegue realizar essas quatro operações na seqüência correta, ela executa qualquer tarefa calculável. O programa estabelece a seqüência correta do computador para determinada tarefa. É onde entram os programadores; e, se a CPU tivesse a capacidade de gerar suas próprias seqüências de operações, eles não seriam necessários.

Para tornar isso mais claro, consideremos o seguinte: todos os programas incluem variáveis, isto é, os nomes das posições da memória em que são armazenadas as informações. A maioria dos programas executa algum tipo de operação aritmética com essas variáveis.

Após tal operação, o programa muitas vezes compara duas unidades de informação e, como resultado, executa um conjunto ou outro de instruções. Em geral, as informações são introduzidas num programa por meio do teclado, e apresentadas para o usuário no vídeo.

Com exceção da frase sobre dispositivos de entrada e saída, essa descrição não contém nada além das quatro operações elementares apresentadas em termos diferentes. Supondo (ao menos por enquanto) que, para a CPU, todos os dispositivos de entrada/saída sejam áreas especiais da memória, então a imagem do computador ideal está completa.

A memória do circuito

Imagine um circuito elétrico simples, composto de uma bateria, um interruptor e uma lâmpada: se ligamos o interruptor, a luz acende, assim permanecendo até que acabe a bateria ou seja desligado o interruptor. O estado da lâmpada — ligado (on) ou desligado (off) — é uma unidade de informação registrada num dispositivo de memória: o próprio circuito.

Suponha que o interruptor tenha sido instalado na entrada de uma fábrica e a lâmpada no escritório do gerente. Ao chegar para o trabalho, o primeiro funcionário liga o interruptor. O gerente fica então sabendo que alguém começou o expediente, pois a luz acende no escritório.

Não é preciso que o gerente esteja presente no momento em que a luz acende. Ele poderá olhar para a lâmpada quando quiser, para verificar se alguém chegou. A informação fica registrada no circuito.

A memória do computador armazena informações de modo análogo: todas elas se resumem à presença ou ausência de eletricidade num circuito. Naturalmente o processo é um pouco mais complexo e, para entendê-lo, precisamos aperfeiçoar o sistema de informações para a gerência. Em vez de um, imaginemos quatro circuitos interruptor/lâmpada (uma seqüência de quatro interruptores na portaria e outra de quatro lâmpadas no escritório).

Um interruptor, ao ser ligado, acende a lâmpada correspondente. Cada empregado, então, aciona os interruptores numa combinação específica: ao chegar, Nancy liga o primeiro e o segundo interruptores, deixando desligados o terceiro e o quarto; Fernando aciona apenas o quarto; Roberto, o primeiro e o terceiro; e assim por diante. As luzes no escritório indicam ao gerente não apenas que alguém chegou para trabalhar, mas também informam quem.

Suponhamos agora que a posição "desligado" esteja associada ao 0, e a posição "ligado" ao 1. Assim, Nancy coloca os interruptores em 1100 (os dois primeiros ligados, o terceiro e o quarto desligados); Fernando forma a combinação 0001 (só o quarto interruptor ligado).

Se o gerente também associar 1 às lâmpadas acesas e 0 às desligadas, tanto ele quanto os funcionários estarão usando a mesma linguagem de identificação. Para ambos 0001 significa Fernando.

Quantas combinações são possíveis no caso? Cada interruptor pode estar em uma de duas posições e há quatro interruptores. Serão, portanto, 2x2x2x2 = 16 combinações diferentes. Examinemos todas as possibilidades:

0000 0001 0010 0011
0100 0101 0110 0111
1000 1001 1010 1011
1100 1101 1110 1111

Por mais que tente, você não conseguirá formar mais que dezesseis combinações.

Observe com que rapidez passamos da imagem concreta de lâmpadas numa sala para a questão abstrata das combinações de 1 e 0. Abstraindo um pouco mais, poderemos transformar essas combinações em números.

Equivalência
A cada funcionário desta hipotética empresa está associada uma configuração específica de lâmpadas acesas e apagadas, que pode ser transcrita de modo imediato em números binários. Estes, por sua vez, podem ser convertidos ao sistema decimal

Nome	n.° decimal	n.° binário
Nancy	12	1100
Fernando	1	0001
Roberto	10	1010

Sistemas de numeração

Pense no processo que empregamos para contar e escrever simultaneamente. Você consegue contar de zero a nove com facilidade porque cada um desses números possui nome e símbolo específicos para representá-lo. Mas o que você escreve depois de nove? Existe um nome (dez) para esse número, mas não um único símbolo para sua representação.

Você é obrigado a reutilizar alguns dos outros símbolos: 10, 11, 12 e assim por diante, até 99, quando novamente se esgotam as combinações possíveis com duas colunas, de modo que o número seguinte deverá ter três colunas (100).

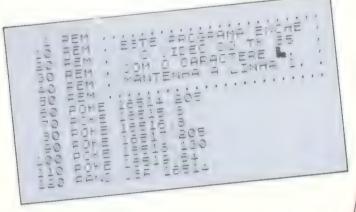
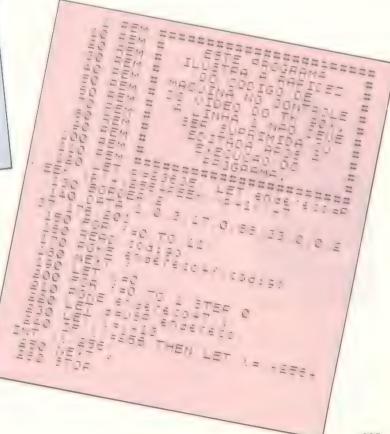
Se isso lhe parece óbvio demais, lembre-se de como foi difícil aprender a contar na escola: todo aquele papel quadriculado e as casas das centenas, dezenas e unidades escritas sobre cada soma! Hoje você sabe que o número 152 significa "1 nas centenas, 5 nas dezenas, 2 nas unidades", ou que 100 + 50 + 2 = 152. A contagem funciona dessa forma porque usamos apenas dez dígitos (0, 1, 2, 3, ..., 9) para representar todos os números possíveis.

Como funcionará então a contagem se tivermos apenas dois dígitos (0 e 1)? Contar até 1 não tem mistério; mas como representaremos o número seguinte? Não dispondo de outros dígitos teremos, portanto, de reutilizar os dois existentes (tal como fizemos ao contar com dez dígitos); e escrevemos o número seguinte como 10.

Sabemos que o número seguinte no sistema decimal chama-se "dois"; portanto, no sistema binário, 10 representa o número 2. O número seguinte na contagem é 3, que representaremos por 11. E depois, o que acontece? Esgotadas as combinações de dois dígitos, representamos o número 4 como 100; o 5 como 101, o 6 como 110 e o 7 como 111. Desse modo, continuamos a contar em números decimais (0, 1, 2 etc.), só que estamos escrevendo em números binários (0, 1, 10, 11, 100, 101, ...).

Da mesma forma que o número decimal 152 equivale a (1x100) + (5x10) + (2x1), o número binário 101 significa (1x4) + (0x2) + (1x1). Em vez de termos as colunas das centenas, dezenas e unidades para esses números, usamos as colunas dos 4, dos 2 e dos 1. Num número decimal, o valor de um dígito multiplica-se por 10 a cada coluna que ele se move para a esquerda; já um número binário é multiplicado por 2.

O sistema binário não passa de uma forma diferente de representar os números. Conhecendo os numerais romanos, você não estranha ao ver um antigo relógio de parede apontar VII horas; e por que não escrever 111 horas? O número efetivo de horas não se altera só porque o representamos de outra maneira.

No entanto, convém chamar o número binário de "um-um-um binário" e escrevê-lo como "111b" para não confundir com uma representação decimal.

Alta velocidade
Estes dois pequenos programas para o TK 85 e o TK 90X, embora escritos em BASIC, ilustram a grande vantagem na programação em código de máquina sobre as linguagens de alto nível: o processamento é muito mais rápido.

PSION

Série integrada
O Xchange, da Psion, é um conjunto de programas integrados com base no software criado para o Sinclair QL. O sistema inclui o processador de texto Quill, o banco de dados Archive, a planilha financeira Abacus e o gerador de gráficos Easel. Os quatro pacotes são comercializados juntos ou em separado. O Xchange foi preparado com versões para os micros IBM PC e PC XT, Apricot e Apricot XI da ACT e Sirius I. Planejam-se versões para o Macintosh da Apple e para o Digital Rainbow.

A Psion consolidou sua imagem produzindo aplicativos para a Sinclair Research. Contudo, numa iniciativa pioneira, surpreende o mercado ao anunciar seu primeiro micro portátil, o Organiser.

Fundada em 1981 por David Potter, o primeiro lançamento da companhia inglesa Psion foi um conjunto de programas para o então recém-lançado micro ZX81, da Sinclair: Flight Simulator, Backgammon, Vu-Calc e Vu-File. Essa pequena série de softwares, composta por um programa simulador de vôo, um jogo de gamão, uma planilha e um banco de dados, tornou o nome da empresa conhecido da noite para o dia. Por isso, um ano depois, não houve surpresa quando a Sinclair escolheu a Psion para desenvolver um cassete de demonstração para o Spectrum. Graças à enorme popularidade dos equipamentos Sinclair, foi rápida a disseminação dos produtos Psion.

Alguns sucessos da companhia são notáveis, como a venda das versões do Flight Simulator para os equipamentos ZX81 e Spectrum, que ultrapassou as 500.000 cópias. Segundo cálculos de mercado, o total de venda mundial dos cassetes da Psion excedeu os 3 milhões. A partir do lançamento da série Xchange, a Psion deu início a um programa de diversificação de seus produtos. Para isso investiu cerca de 2 milhões de libras, garantindo sua participação no mercado de softwares profissionais.

A Psion sempre se caracterizou por uma abordagem inovadora. No desenvolvimento de seus softwares para micros, procurou usar técnicas avançadas como a "compilação cruzada", que permite a elaboração, em determinado equipamento, de um programa que será rodado em outra máquina. Assim, o Horizons, destinado ao Spectrum, foi desenvolvido num TRS-80.

Atualmente a empresa utiliza dois minicomputadores VAX 750 para criação de seus softwares. Nesses equipamentos, foi desenvolvida a série de programas para o micro QL: a planilha Abacus, o banco de dados Archive, o gerador de gráficos Easel e o processador de textos Quill. Por sua vez, a Psion Support organizou a QLUB, uma linha de apoio aos usuários do QL, respondendo por escrito às dúvidas no prazo de 48 horas.

Estuda-se também a criação de sistemas profissionais para os computadores IBM PC, Apricot, Sirius, DEC Rainbow e Macintosh. A mesma estrutura da série de aplicativos para o QL repete-se na série Xchange, que se compõe de uma planilha, um banco de dados, um gerador de gráficos e um processador de textos. Esses dois pacotes de software se destacam dos concorrentes pela facilidade com que os dados podem ser transferidos de um para outro.

Um produto da companhia que também chamou muita atenção foi o microcomputador portátil Organiser. Para o relações-públicas da empresa, Robin Kinnear, o fato se explica justamente por se tratar de uma empresa especializada em software: "O Organiser é um projeto muito atraente em termos de compactação de software, e não havia nada parecido no mercado". Assim, a Psion decidiu fabricar seu próprio hardware, influenciada pela grande experiência com software.

Para o Organiser são produzidos três aplicativos — de ciências, matemática e finanças —, juntamente com módulos de RAM de 8 Kbytes e de 16 Kbytes, os chamados "datapacks".

A Psion avalia também a possibilidade de acrescentar outros pacotes de programas e, nesse sentido, vários projetistas se mostraram interessados em desenvolver software para o Organiser.

A outra prioridade da Psion se refere a sua expansão no mercado internacional. A empresa mantém subsidiárias nos Estados Unidos e na África do Sul, pretendendo também garantir a presença de seus produtos na Europa mediante contratos com distribuidoras locais. Além disso, a ligação com a Sinclair contribui para os planos da Psion de conquistar o mercado da Europa oriental. O primeiro passo nessa direção foi dado pela exportação de quatrocentas unidades do ZX81 para a Tchecoslováquia.

Da teoria à prática
A Psion tem como fundador e sócio majoritário David Potter, especialista em Física Computacional pelo Imperial College, de Londres, e pela Universidade da Califórnia. A empresa chamava-se antes Potter Scientific Investments.

MIDI NA PRÁTICA

Dezenas de empresas têm produzido, no mundo inteiro, interfaces digitais para instrumentos musicais. Este artigo vai mostrar alguns dos modernos pacotes de hardware e software.

O Micon (MIDI controller, "controlador de interface digital para instrumentos musicais"), produzido pela firma inglesa XRI Systems, é um dos típicos pacotes das pequenas empresas de software. Destina-se ao uso com o Sinclair Spectrum (TK 90X no Brasil) e compõe-se da própria interface e de um software em fita cassete. Seu seqüenciador tem oito pistas independentes, cada uma com capacidade para 2.951 eventos (notas, pausas etc.). Introduz-se a música por meio do teclado do sintetizador e, com a tecla de espaços do micro, define-se o número de eventos a serem executados.

Quando se usa o sintetizador, cada nota passa pela tela sob a forma de grosseira notação padrão de pentagrama, apresentando claves de fá (baixos) e de sol (médios e agudos), amplitude total de durações (inclusive pontos de aumento), sustenidos, bemóis e indicadores de staccato (nota de articulação curta). A notação do vídeo não é muito boa para indicar pausas, e as hastes de mínimas a semifusas sempre apontam para cima, em vez de alternar o sentido conforme sua posição no pentagrama. Além disso, as seqüências de notas curtas não são reunidas em grupos que expressem a métrica ou apontem o andamento da música. O resultado, embora difícil de seguir, exceto como guia de referência, pode ser impresso.

A vantagem dessa apresentação fica evidente quando se vai editar a composição. A música é dividida em compassos e, quando se introduz o número de um deles, podem-se suprimir, inserir ou alterar notas. Outros recursos permitem eliminar compassos inteiros, repetir grupos deles em qualquer trecho da seqüência ou, ainda, gravar em fita até dez seqüências (compreendendo quase 24.000 notas), juntamente com as características desejadas de reprodução. O ritmo, por sua vez, pode ser predefinido dentro de uma faixa de 4 milissegundos, ou então controlado por um computador rítmico com comando de sincronização conectado à entrada da interface.

O Micon dispõe de recursos de seqüenciação em tempo real. Na verdade, ele "ouve" a execução num sintetizador e introduz todos os dados na memória do Spectrum. Não reproduz uma partitura da execução, mas se mostra útil de duas outras maneiras: primeiro, permite a audição pelos tecladistas daquilo que executaram sem que precisem se preocupar com ajustes de níveis de entrada na fita (essa resposta imediata é particularmente importante na educação musical). Em segundo lugar, uma vez que a música desejada já esteja "nas pontas dos dedos" de quem a executa, alivia bastante o tédio da entrada passo a passo. Entretanto, o músico ainda necessita de um metrônomo ou de um padrão de bateria eletrônica para manter o ritmo uniforme, do contrário a seqüência será reproduzida com todos os erros de métrica que ele cometer. Como em qualquer sistema MIDI, os resultados obtidos com o Micon dependem da sofisticação do sintetizador ao qual a interface está ligada.

O pacote Jellinghaus Music System custa um pouco menos que o Micon e destina-se a uma faixa mais ampla de máquinas: o Apple II, o Commodore 64 e também o ZX Spectrum. Sua operação assemelha-se à do Micon, já que as informações musicais são introduzidas pelo teclado do sintetizador interfaceado. A principal diferença é que isso ocorre apenas em tempo real, e a apresentação da música no vídeo não usa o pentagrama convencional.

A organização da música é feita em compassos, indicados pelo número de semínimas ou colcheias em cada grupo. Utilizam-se quatro tipos de compasso: com três, quatro ou cinco semínimas (3/4, 4/4 ou 5/4), ou sete colcheias (7/8) e, como no Micon, o ritmo de execução tem de

Controle total

O microcomputador Commodore 64 é um dos mais versáteis para se lidar com sons. Pode-se criar um programa em BASIC para dar entrada às notas com um joystick. Neste exemplo, o ponto luminoso que aparece sobre o teclado musical pode ser movimentado para cima e para baixo na escala, levando-se a alavanca do joystick respectivamente para a frente e para trás. Uma vez localizada a nota certa, sua duração é selecionada movendo-se a alavanca para os lados. Grava-se a nota apertando o botão de disparo. Pode-se, então, proceder à escolha da nota seguinte.

1 Controle de computador

ser coordenado com um metrônomo ou um computador rítmico.

Se a execução em tempo real se desviar do ritmo, os dados de entrada poderão tornar-se uma cadeia de notas "descompassadas". Uma vez ajustada a seqüência para reprodução, no entanto, não haverá problemas — até que apareça um agrupamento cujo compasso não seja 3/4, 4/4, 5/4 ou 7/8. Se isso ocorrer, o seqüenciador falhará. Significa também que um compasso como 12/8 não pode ser usado, mesmo que o ritmo seja perfeito: uma omissão importante, já que 12/8 é um compasso usado com freqüência em jazz, rock, funk e reggae.

Os dados de uma seqüência aparecem no vídeo sob a forma de colunas, mostrando a oitava na qual a nota ocorre, sua intensidade, duração (em semínimas e colcheias), tempo de retardo e as teclas de comando para tirar sons no computador (de 0 a 9).

Esse tipo de apresentação proporciona um leque bem mais amplo de informações do que a notação padronizada e resumida existente no Micon, que não incorpora tempos de retardo ou níveis de andamento — pelo menos em forma numérica, que somente entrou no vocabulário musical com o advento de recursos sonoros nos equipamentos eletrônicos digitais.

Entretanto, muitos músicos sem treinamento formal logo desenvolvem a habilidade de olhar um trecho de pentagrama comum e, pelo desenho ascendente ou descendente das notas, ter uma idéia aproximada de como soará uma melodia ou um acorde.

Significa que, com a prática, a notação pode ser lida, e uma seqüência musical "ouvida" mentalmente em tempo real. A apresentação em colunas pode ser excelente para verificação dos dados, mas poucas pessoas conseguiriam solfejar uma melodia a partir dela. Um sistema ideal incluiria ambos os tipos de representação; mas, na falta dele, a notação convencional do Micon é mais eficiente e adequada à educação musical.

O principal problema para o usuário de micro que pretenda usá-lo na música não será a escolha do pacote MIDI adequado, mas sim a obtenção de um sintetizador sofisticado o bastante para extrair o máximo da interface, sem que precise hipotecar a casa para pagá-lo. Quem quiser um sintetizador capaz de dar conta de 50% das trilhas complementares de uma canção popular típica verá que o custo de tais instrumentos situa-se em torno de 250 dólares. Abaixo disso, a maioria dos sintetizadores dispõe de recursos limitados demais.

O *sintetizador conceitual* representa outra possibilidade: trata-se de um pacote de software avançado e um periférico com teclado musical que utiliza a geração de som de um microcomputador. É uma opção melhor do que interfacear com um sintetizador projetado antes do lançamento da MIDI.

Um dispositivo desse tipo é o PDSG (Programmable Digital Sound Generator, "gerador

2 Sintetizador

3 Interface

de som digital programável"), fabricado pela Clef Products na Inglaterra, para uso no BBC Micro Modelo B. Seu teclado musical sensível, de 61 notas e dois pedais, à base de variação da velocidade de acionamento, confere dinâmica à execução. Assim, os dados subjetivos da interpretação em tempo real podem ser armazenados para gravação e reprodução.

O PDSG tem 32 unidades digitais geradoras de som, em vez de osciladores, e cada uma abrange até onze características definidas. Se necessário, as 32 são usadas para produzir uma única nota. Apenas esse recurso nas mãos de um usuário talentoso dá ao PDSG riqueza e variedade de sons ao mesmo nível da maioria dos sintetizadores relacionados no box.

Se uma única unidade de geração for designada para cada nota, uma seqüência programada poderá incluir 32 linhas diferentes. Ou, então, uma parte dos geradores poderá ser usada para o material seqüenciado, e o restante tocado em tempo real simultaneamente. As características sonoras, em forma de ondas, são apresentadas no vídeo, dando a oportunidade de análise visual dos sons — um recurso inestimável e que torna o PDSG apropriado à educação musical. A unidade geradora de som — adequada para seqüências lineares "passo a passo", geração de formas de ondas e análise espectral — custa por volta de 350 dólares, o mesmo preço do teclado.

A principal desvantagem do PDSG encontra-se na representação dos sons no estágio de conversão digital-analógico. O ouvido e o cérebro interpretam sons com freqüências entre 20 Hz e 20 kHz. Os sons naturais, inclusive os produzidos por instrumentos musicais acústicos, atuam em toda essa faixa e mesmo além dela. O PDSG, porém, reproduz sons apenas com freqüências de até 12 kHz. Como resultado, sua qualidade compara-se à de um sistema doméstico de alta fidelidade (os fabricantes supõem que um amplificador e alto-falantes serão usados para completar o sistema).

A interface MIDI foi pioneira, ao permitir o acesso de usuários de micros a sintetizadores de tempo real. O sistema PDSG pode encorajar os que já possuem um sintetizador a optarem pela compra de um micro e um sintetizador conceitual.

Queda para a música

O desenvolvimento da interface MIDI proporciona aos proprietários de microcomputadores uma ampla gama de possibilidades para composição musical.

Ao mesmo tempo, há o risco de se comprar um pacote caro apenas para se emaranhar na complexidade do sistema.

Uma alternativa é começar com um sistema musical mais barato, para se familiarizar com os rudimentos da música eletrônica. Tal sistema deve ter boa qualidade para ser musicalmente satisfatório e estimular o interesse pelas possibilidades mais amplas dessa categoria musical. Um bom pacote inicial seria o Ultisynth 64, em cassete, produzido pela Quicksilva, que explora o chip SID (Sound Interface Device, "dispositivo de interface sonora") do Commodore 64 e seus três osciladores.

Usando esse pacote, cada tecla do Commodore torna-se um controle independente para gerar e definir sons. Os quatro formatos básicos de onda — senóide, quadrada, triangular e dente de serra — podem ser usados em combinação com vários ajustes. Filtram-se os sons para que sejam melhor caracterizados. Também está incluida a modulação Ring — processo que dá a soma e a diferença de duas freqüências quaisquer —, muito útil para se produzirem sons com timbres metálicos, como sinos e gongos. Além disso, é possível gravar ritmos preestabelecidos e seqüenciar até 2.048 notas.

Os recursos do Ultisynth assemelham-se bastante aos do VCS 3, um sintetizador "clássico" controlado por variação de voltagem, do final dos anos 60 e início dos 70. Hoje, o pacote Ultisynth é relativamente barato e ideal para uma introdução à síntese musical.

Outro sistema em cassete adequado para o principiante é o Multisound, da Romik, bastante parecido com o Ultisynth nos recursos de controle, mas que apresenta no vídeo um teclado musical. Escolhem-se as posições nesse teclado por meio de um cursor e define-se a duração das notas pelo teclado do computador.

REFERÊNCIAS CRUZADAS

Certos tipos de arquivo requerem diferentes abordagens para organização, inter-relacionamento e pesquisa de seus dados. Veremos de que maneira os SGBDs podem nos auxiliar nessa tarefa.

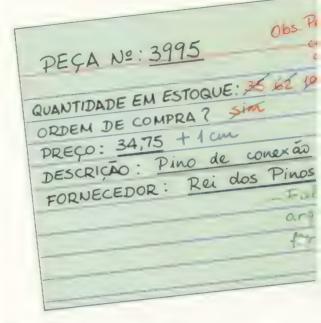
Os registros contidos num banco de dados simples são absolutamente auto-suficientes. Em outras palavras, todo registro inclui informações completas, tornando dispensável a referência a outro registro. É o caso dos tradicionais arquivos que utilizam fichas de cartolina. Temos outro exemplo de banco de dados simples no fichário de uma biblioteca: cada registro consiste apenas nos campos TITULO, AUTOR, EDITOR e um número padronizado de catalogação, como o CIP da Câmara Brasileira do Livro ou o ISBN (International Standard Book Number, "número padrão internacional de livro").

Algumas vezes um campo contém um item de dados capaz de constituir outro registro. Num arquivo de sócios de um clube, por exemplo, o campo FILHOS pode referir-se a outros registros do mesmo arquivo, caso os filhos de um associado também sejam membros do clube.

No entanto, um campo às vezes remete registros que não se ajustam à estrutura do arquivo de um banco de dados. Considere um arquivo de dados para o inventário de peças, contendo os campos NUMERO DA PECA, PRECO, UNIDADES EM ESTOQUE e FORNECEDOR. É bastante provável, nesse caso, que o campo FORNECEDOR também se refira a outros registros que não se adaptam à estrutura definida para o banco de dados PECAS.

Como ilustração, vamos começar pelo arquivo SOCIOS:

NOME DO SOCIO	Sueli Gomes
ANO DE ADMISSAO	1979
JOIA	Quitada
PROFISSAO	Professora
RENDA MENSAL	2890000
FILHOS	Julia Gomes, Filipe Gomes

Nesse exemplo, o campo FILHOS compreende dois itens de dados; ambos poderiam ter registros semelhantes no arquivo SOCIOS se também fossem sócios. Compare com um registro do banco de dados PECAS:

NUMERO DA PECA	3995
UNIDADE EM ESTOQUE	86
PRECO	US$ 34.75
DESCRICAO	Pino de conexao
FORNECEDOR	Rei dos Pinos Ferragens Ltda

Existem duas informações no campo FORNECEDOR; nenhuma delas teria regis.ros compatíveis com a estrutura do banco de dados PECAS. Portanto, um inventário de peças desse tipo exige um segundo arquivo para os fornecedores, no qual cada registro seria mais ou menos assim:

FORNECEDOR	Importex Ltda
ENDERECO	Rua do Bosque, 97 Sao Paulo, SP
TELEFONE	(011)227-4391
ARTIGO1	Pino de conexao
PRECO1($US)	34.75
ARTIGO2	Oleo (meio litro
PRECO2($US)	6.00
ARTIGO3	Polidor (lata pequena)
PRECO3($US)	2.30
ARTIGO4	Polidor (lata grande)
PRECO4($US)	4.00
ARTIGO5	Soquete para pino
PRECO5($US)	7.40
ARTIGO6	—

As informações a respeito de cada fornecedor têm estrutura inteiramente diversa das referentes às peças vendidas por eles. A solução é montar dois arquivos diferentes — um para as peças e outro, separado, para cada fornecedor.

Organização de estoque
Os arquivos de fichas podem ser tão complexos quanto o usuário deseja. Os SGBDs simples manipulam os dados de forma estruturada, e os avançados permitem inter-relacionar arquivos, registros e campos. Na ficha da ilustração, o campo ORDEM DE COMPRA só será relevante quando o estoque se aproximar de zero. Por isso, alguns SGBDs possibilitam ao usuário montar campos cuja existência depende do conteúdo de outros campos.

Manipulação múltipla

Para implementar de modo eficiente arquivos diferentes mas inter-relacionados, o SGBD ("sistema gerenciador de banco de dados") deverá ser capaz de manipular mais de um arquivo de cada vez. Os SGBDs menos sofisticados, como o Card Box, operam um único arquivo de cada vez; mas outros, como o dBase II, têm a possibilidade de usar um arquivo primário (como PECAS) e outro arquivo secundário (como FORNECEDOR).

Um SGBD para aplicações profissionais permite a classificação de um item num arquivo primário (para extração de registros selecionados), assim como permite a pesquisa de registros relevantes (como FORNECEDOR) em arquivos relacionados.

No software dBase II, é possível o uso simultâneo de dois arquivos inter-relacionados por meio do emprego de campos com chaves comuns. Dois comandos possibilitam o estabelecimento da relação de um para outro: SELECT PRIMARY e SELECT SECONDARY. Com PECAS como arquivo primário e FORNECEDOR como secundário, processamos o primeiro por meio do comando USE PECAS. Para obter a referência cruzada de fornecedores, utilizamos SELECT SECONDARY e, na linha seguinte, USE FORNECEDOR. Todos os comandos normais do dBase II ficam disponíveis e processam o último arquivo selecionado.

Há ocasiões em que o conteúdo de um campo (como no caso de FORNECEDOR, no arquivo PECAS acima mencionado) não exige a flexibilidade de registros completamente dependentes (em arquivos diferentes) e, portanto, campos dependentes num mesmo registro seriam suficientes.

Como exemplo, suponha que alguém queira montar um banco de dados sobre as últimas conquistas tecnológicas da indústria automobilística. Com base em recortes de jornais, livros, programas de tevê etc., relacionaria itens tão diferentes como carrocerias de plástico moldadas por injeção e injetores de combustível controlados eletronicamente.

Um dos formatos possíveis para os registros desse banco de dados seria o seguinte:

TEMA: ______
SINOPSE1: ______
SINOPSE2: ______
SINOPSE3: ______
SINOPSE4: ______
SINOPSE5: ______
FONTE: ______ ISBN: ______
CIP: ______
TITULO: ______
DATA: ______
PAGINA N: ______

Se a fonte de um registro for uma revista, o campo ISBN ou CIP será irrelevante, já que revistas não o trazem. Já no caso de ser um livro — por exemplo, o *Programming ROM chips for in-car computers* —, haveria o ISBN.

Analogamente, seja a fonte um livro ou uma revista, haverá um número de página. Contudo, quando a fonte for uma transmissão de corrida de Fórmula 1 pela televisão, o campo PAGINA N. não será necessário. Certos SGBDs permitem que se calcule a presença ou ausência de determinado campo em função de outros campos anteriores.

Assim, quando FONTE não for um livro, o campo ISBN não se mostrará para digitação; tampouco surgirá no vídeo ou impresso. E, também, quando não for nem livro nem revista, o campo PAGINA N. não será chamado. Um SGBD que faz isso economiza espaço na memória ao eliminar campos desnecessários e ajuda a melhorar a apresentação das informações. Ainda assim, contudo, é menos versátil que o SGBD capaz de inter-relacionar registros de dois arquivos distintos.

Nos bancos de dados contendo informações de base que permanecem constantes em todo o registro e informações secundárias que podem ou não ser necessárias, ocorre o que é conhecido como "hierarquia em dois níveis".

Como regra geral, SGBDs de múltiplos arquivos processam dados hierarquizados em dois níveis, como um subconjunto ou um caso especial de um banco de dados de múltiplos arquivos. Por exemplo, uma referência FONTE a um livro poderia relacionar-se com um arquivo LIVROS, e uma referência FONTE a um programa poderia relacionar-se com um arquivo PROGRAMAS.

Um arquivo de fichas escritas à mão pode ser tão complexo quanto se queira. Contudo, há sérias limitações entre os diferentes SGBDs disponíveis no mercado, quanto ao que podem ou não gerenciar estruturalmente. Assim, antes de comprar um SGBD para seu micro, pense bastante no que, precisamente, você quer que ele faça, além de pesquisar com todo cuidado as especificações dos softwares à venda.

TOQUE DE MESTRE

O tablete gráfico Touchmaster é utilizável com a maioria dos microcomputadores mais populares da atualidade. Pode ser empregado também como um teclado simplificado.

Os computadores de maior sucesso da atualidade comportam geradores de gráficos de alta resolução. No entanto, a menos que se disponha de software para desenho, será necessário muito tempo e esforço para criar tais gráficos e muitos recursos ficarão sem plena utilização. Um programa para desenho à mão livre não basta, pois muitas vezes o usuário deseja copiar no computador uma imagem existente.

Diversos tabletes gráficos foram comercializados com esse propósito, mas, em sua maior parte, restringiam-se a determinados micros.

O tablete Touchmaster
Selecionando o comando apropriado, no lado direito do aparelho, e movimentando o lápis especial (sem grafite) ou mesmo o dedo na área para desenho, o comando será executado na tela.
Esse tablete gráfico é compatível com muitos dos micros mais populares

O Touchmaster foi desenvolvido para operar com larga faixa de microcomputadores (alguns exigindo interface ou cabo adequados). Saídas serial e paralela são fornecidas para permitir esse uso com vários micros ou com o próprio teclado do Touchmaster.

Aliás, esse tablete está sendo anunciado como um teclado substitutivo, mas a simplicidade de seu projeto significa que tal uso se restringe à seleção entre opções de menu ou a um simples controle de jogos. Exige-se ainda teclado de computador para entrada de dados, bem como para carregar o próprio software do Touchmaster.

O equipamento vem instalado num estojo cinza bem-acabado, medindo 350 x 330 x 35 mm. A parte posterior, um pouco elevada, forma um ângulo conveniente ao desenho. No transformador que acompanha o tablete gráfico, um LED vermelho indica quando o aparelho está ligado; no entanto, ele não dispõe de interruptor liga/desliga.

O Touchmaster conta com a tecnologia de membrana, desenvolvida nos teclados dos microcomputadores ZX81 e Spectrum, e fornece uma resolução de 256 x 256 pixels, bem menor que a proporcionada pela grande maioria das telas de vídeo. Uma malha isolante separa as camadas superior e inferior (resistência); exercendo-se pressão sobre a camada superior, a malha entra em contato com a película.

Como o tablete contém um microprocessador que faz a varredura da película superior num sentido e da inferior em outro, a coordenada do ponto de contato é enviada às saídas serial e paralela.

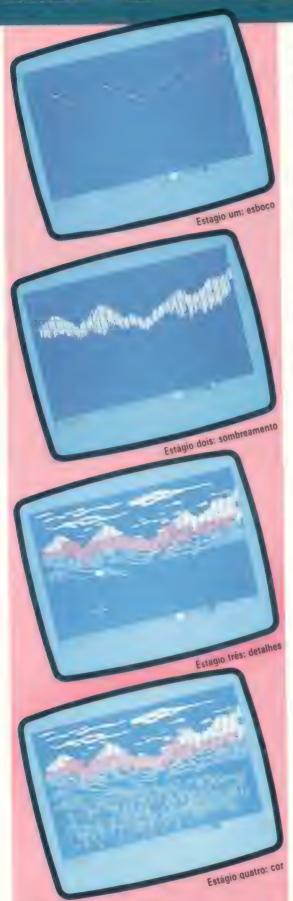
A saída serial é utilizada para conectar o tablete às interfaces RS232C, ao passo que a paralela destina-se ao uso com os micros TK 85, CP 200, Commodore e outros.

Os manuais são bastante inadequados: o de hardware instrui sobre a conexão do aparelho e fornece vários programas simples em BASIC, para leitura de coordenadas, mas com detalhes insuficientes.

Programa Multipaint

Com o Touchmaster, vem o programa Multipaint para desenho, que demonstra os recursos oferecidos, mas não é um gerador de gráficos abrangente.

Um gabarito de plástico fornece um menu dos recursos disponíveis, com a opção selecionada exibida numa janela de status (estado) na parte inferior da tela. Podem ser usados cinco diferen-

tes tipos de pincel (brush), cada um com largura entre 2 e 32 pixels, em passos de 2 pixels. A janela exibe também o modo corrente de desenho — pontos, pontinhos ou à mão livre — e as cores selecionadas para fundo e primeiro plano. As cores mudam conforme se pressiona no menu a opção indicada, até que a escolha desejada apareça na janela de status.

Uma vez selecionadas as cores corretas e também os tipos de pincel, opções adicionais estão à disposição para a criação de retângulos, círculos, polígonos e anéis. O pacote oferece um lápis apropriado, mas também pode-se usar a pressão dos dedos.

Com sua ampla área de trabalho, o Touchmaster não sofre as restrições apresentadas por outros tabletes gráficos, como o Koala: a pressão de um dedo será traduzida em coordenadas precisas, e não em mera aproximação.

Contudo, o Multipaint oferece apenas recursos rudimentares. Uma opção fill (preencher) vem marcada no gabarito e documentada nos manuais, mas — pelo menos na versão Spectrum — não opera da maneira esperada. Também não se acha presente qualquer recurso para ampliação ou edição, significando que não se pode variar as cores. No Spectrum, onde com freqüência é mais fácil desenhar em preto e branco antes de adicionar cor, essa desvantagem fica mais do que evidente.

Por outro lado, o Touchmaster Pad possui construção robusta e oferece grande área para desenho, dimensionada no padrão A4. Outra vantagem significativa está no fato de que, se você decidir modernizar seu micro com recursos mais atualizados ou comprar outro, pode continuar utilizando esse tablete gráfico mediante o uso de nova interface e software apropriado. Infelizmente, é desapontador que a documentação fornecida pelo fabricante e o software disponível sejam tão precários em confronto com o próprio padrão de qualidade desse tablete gráfico.

Uma cena tocante

O Touchmaster pode ser utilizado para produzir gráficos, aliado ao software Multipaint e aos modelos gráficos oferecidos com o produto. As figuras mostram os estágios de desenvolvimento de uma cena, do esboço à imagem colorida e sombreada.

Imaginação à solta

Há vários jogos, programas educacionais e outros utilitários disponíveis para o Touchmaster. Cada pacote inclui software apropriado (para vários micros), instruções e imaginativos modelos gráficos.

PONTA-DE-LANÇA

Desde seu lançamento, a série de microprocessadores Motorola 68000, desenvolvidos a partir da CPU 6809, conquistou significativa parcela do mercado de processadores de 16 e 32 bits.

O 68000 guarda forte semelhança com o MC6800, antigo microprocessador Motorola ainda muito utilizado, particularmente em periféricos. Bem mais potente, o 68000 tem apoio de ampla variedade de hardware, como placas de entrada/saída com o chip 6821 PIA (Peripheral Interface Adaptor, "adaptador de interface periférica"); unidades de vídeo com CTRCs 6845 (Cathode Ray Tube Controllers, "controladores de tubo de raios catódicos"); relógios que usam o temporizador programável 6840; e controladores de discos.

Outra característica do 68000 que o torna bastante atraente para os projetistas de computadores é a largura dos barramentos (bus) de dados e endereços. Estes se acham separados, e cada bit tem seu próprio pino — o que não ocorre no 8086, no 8088 e no Z8000, todos com pinos compostos: os dois barramentos partilham um conjunto de pinos e os sinais são multiplexados e demultiplexados em seu destino.

Portanto, o processador pode trabalhar tão rapidamente quanto o resto do sistema permitir. Com os modernos chips de RAM de 50 ou 90 nanossegundos (10^{-9} segundo), isso significa uma redução ou mesmo eliminação dos atrasos. O processador mais rápido da série 68000 é o 68000L12, com ciclo de operações de 14 MHz.

No micro Sinclair QL utilizou-se o sucessor do 68000, o 68008. Este, internamente, se parece muito com os outros da série. Mas, a fim de se mostrar mais compatível com os sistemas de 8 bits existentes, o 68008 tem um barramento de dados de 8 bits, em vez dos 16 bits de largura plena. Já que necessita de poucos pinos, utiliza o encapsulamento comum de quarenta pinos.

Em fins de 1984, a Motorola planejava lançar um modelo ainda mais potente: o 68020, de 32 bits, que requer encapsulamento de 96 pinos. Também se encontrava em desenvolvimento o 68881, um processador matemático de ponto (vírgula) flutuante com oito registradores (de 80 bits cada). O 68881 aumentará em muito a quantidade de dados reais processáveis.

Vários outros chips da série 68000 proporcionam funções de entrada e saída melhoradas. Do ponto de vista do programador, no entanto, o 68000 apresenta muitas vantagens sobre à maior parte dos outros processadores de 16 bits, devido à simetria de seus registradores de dados e endereços e ao extenso conjunto de instruções.

Série campeã

6502
Desenvolvido em tecnologia MOS, o microprocessador 6502 foi projetado para se tornar, junto com o Zilog Z80, o esteio da indústria de microcomputadores. Utiliza um bus de endereço de 16 bits e um bus de dados de 8 bits. Sua principal peculiaridade é a organização de registros. Dispõe de apenas um acumulador, mas toda a página zero da memória pode ser usada para registros de emprego geral.

68000
A Motorola reprojetou e aperfeiçoou o 6800 até chegar ao 6809, mas não a tempo de conseguir uma grande fatia do mercado de 8 bits. Em função disso, a companhia desenvolveu os processadores 68000 de 16 e de 32 bits. O 68000 utiliza muitos chips de apoio das séries 6502 e 6800 e foi construído em torno de oito registradores de dados e sete de endereços, todos de 32 bits.

Z80
Tecnicamente mais potente que o MOS 6502, o Zilog Z80 emprega estruturas de bus de dados e endereços semelhantes, mas tem um conjunto de registradores bastante ampliado (doze registradores de uso geral de 8 bits e dois registradores apontadores de 16 bits) e um conjunto de instruções muito mais amplo. Uma de suas maiores vantagens sobre o 6502 é a capacidade de suportar o sistema operacional CP/M.

Contudo, o 68000 não é um microprocessador perfeito. Um de seus problemas reside na distinção que faz entre os registradores de dados e os de endereços, embora sejam de mesma capacidade (32 bits) e, em muitos aspectos, operados pelas mesmas instruções. Como resultado, muitas vezes se faz necessário transferir dados de um registrador de endereços para um de dados, manipulá-los e então devolvê-los ao registrador de endereços. Seria mais prático haver margem para que qualquer registrador fosse utilizado tanto para dados como para endereços.

Há também redundâncias no conjunto de instruções, mas, como resultado do que poderia ser chamado de "modo de endereçamento permutado", isso não se torna um problema. O fenômeno surge porque os vários modos de endereçamento são tão diferentes que, às vezes, um deles se destina à mesma coisa que outro, embora seja acessado por meio de instruções diferentes.

De modo geral, contudo, a série Motorola 68000 proporciona CPUs rápidas, eficientes e de grande capacidade, e que vêm sendo cada vez mais usadas. Seus recursos teriam custado uma pequena fortuna há poucos anos, mas hoje têm um preço razoável. Portanto, é provável que se tornem tão populares entre a próxima geração de máquinas quanto o Z80 e o 6502 são na atual.

CANON T70

Um dos principais problemas para o fotógrafo novato é o excesso de detalhes técnicos do equipamento. A mais recente geração de câmaras, contudo, usa microprocessadores que facilitam o ato de fotografar.

Ao bater uma foto, a primeira tarefa do fotógrafo é encontrar a exposição correta. Isso envolve a determinação da quantidade de luz, proveniente da imagem, que atinge o filme no interior da câmara. Se for demasiada (superexposição), a foto ficará muito clara; do contrário (subexposição), se apresentará muito escura. Para uma exposição correta, é necessário encontrar o equilíbrio entre a abertura do diafragma e a velocidade. Esta determina o tempo de exposição, enquanto aquela diz respeito ao diâmetro do orifício, responsável pela maior ou menor quantidade de luz que chegará ao filme.

Assim, deve-se primeiro medir a quantidade de luz proveniente da imagem; e depois, considerando a sensibilidade do filme, ajustar a abertura e a velocidade. Existem fotômetros que permitem ao fotógrafo uma medição confiável da luminosidade. Muitas câmaras, porém, já vêm equipadas com fotômetro, embora o fotógrafo ainda tenha de escolher uma combinação de velocidade de obturação e abertura que se ajuste a sua leitura.

O desenvolvimento da eletrônica nos anos 70 permitiu converter a leitura do fotômetro em valores de abertura ou de velocidade, sem qualquer intervenção do usuário. Assim, obtêm-se fotos de boa qualidade apenas apontando a câmara para um objeto e pressionando o disparador. Esse é um recurso útil tanto para o principiante como para o profissional, que precisa fotografar em condições difíceis.

A Canon A1, um bom exemplo de câmara eletrônica, permite seis diferentes modos de fotografar, conforme segue.

Prioridade para velocidade. O usuário escolhe a velocidade de obturação, e a câmara determina a abertura correspondente.

Prioridade para abertura. O usuário escolhe a abertura, e a câmara determina a velocidade de obturação.

Programado. A câmara ajusta a velocidade e a abertura, mediante um programa que proporciona a melhor combinação entre ambas.

Flash automático. Existem flashes que, incorporados à câmara, fixam automaticamente a correta velocidade para a operação (1/60 segundo) e ajustam a abertura. Um fotômetro embutido

Câmara pronta
Os microprocessadores passaram a equipar as câmaras na década de 70. Os modelos automáticos de hoje controlam o flash, o tempo de exposição e a abertura do diafragma permitindo, mesmo aos novatos, fotos de alta qualidade em quaisquer condições de iluminação. Duas dessas câmaras, a Nikon FA e a Pentax Super A, processam programas de controle para atender diferentes condições

no flash desliga a corrente deste quando a imagem reflete luz suficiente.

Prioridade para abertura fixada. Serve para objetivas antigas e certos acessórios que trabalham com abertura sempre "fixa" no valor usado para tirar as fotos. As objetivas modernas permanecem com a abertura máxima, a não ser no instante em que a foto é batida, de forma a manter, no visor, a imagem mais luminosa possível.

Manual. Tanto a velocidade como a abertura são ajustadas pelo fotógrafo. Esse sistema permite o controle total da câmara, indispensável nas ocasiões de iluminação deficiente ou quando se pretende algum efeito especial.

Um mostrador eletrônico embutido no visor da Canon A1 indica em que modo a câmara está operando, bem como os valores fixados para velocidade e abertura. O mostrador usa LEDs (Light-Emmitting Diodes, "diodos emissores de luz") que possibilitam sua leitura no escuro.

Apesar da ampla utilidade do modo programado da Canon A1, seu funcionamento é muito simples. Em alguns poucos casos, o programa não determina a melhor combinação possível. Por exemplo, se as fotos forem tomadas ao anoitecer, a câmara poderá escolher uma velocidade de 1/30 segundo e abertura de f2.8. Qualquer foto tirada com velocidade inferior a 1/60 segundo corre o risco de sair "tremida", devido aos leves movimentos da mão do fotógrafo. Assim, quando a câmara é operada nessa faixa de velocidade, um sinal intermitente avisa sobre o risco de se perder a foto; o programa, porém, não seleciona uma abertura maior que permita velocidade de obturação mais elevada.

Imagem do visor
A câmara tipo monorreflex mostra ao fotógrafo a cena através da objetiva

Mostrador LCD de programa
Há três programas de exposição/abertura à escolha do usuário (para objetiva comum, teleobjetiva e grande-angular), além de opções para controle semi-automático e manual

Espelho móvel
Dirige a imagem para o prisma do visor até que o disparador seja pressionado

Concorrência acirrada

Várias empresas lançaram câmaras multimodais com microprocessadores embutidos. A Pentax Super A usa um programa ligeiramente mais sofisticado que o da Canon A1, o que proporciona melhor combinação de abertura e velocidade para as diversas condições de iluminação. Numa foto ao anoitecer, como a do exemplo acima, a Pentax Super A escolheria velocidade pouca coisa inferior a 1/60 segundo (tempo de exposição maior) reduzindo a possibilidade de tremer. A Nikon FA dispõe de um programa alternativo, ativado quando se usa teleobjetiva (com distância focal de 135 mm ou mais). O programa, otimizado para evitar a tremida com objetivas maiores, usa velocidades e aberturas maiores.

Na esteira da Nikon, a Canon colocou três programas alternativos em seu modelo T70. Um deles se destina a objetivas comuns, outro a teleobjetivas e o terceiro a grande-angulares. No entanto, ao contrário da Nikon FA, a câmara não identifica automaticamente a objetiva incorporada, cabendo ao usuário a escolha do programa apropriado. Não se trata, nesse caso, de uma desvantagem, uma vez que deixa ao fotógrafo maiores possibilidades criativas. Por exemplo, para fotografar com grande-angular um tema em movimento rápido, pode-se escolher o modo de teleobjetiva, a fim de conseguir maiores velocidades de obturação.

Um dos grandes problemas das câmaras automáticas reside na leitura do fotômetro. Esse dispositivo indica a exposição média para a imagem toda, podendo ser "enganado" por uma área de extrema luminosidade. Na foto de uma motocicleta contra um pôr-do-sol, por exemplo, o fotômetro indicará a exposição correta para o sol, deixando a moto muito escura. Já ao fotografar a moto contra um fundo escuro, a câmara tratará o tema como se fosse muito mais escuro, e o resultado será uma foto superexposta.

A Nikon FA usa um método original para contornar o problema. Em vez de ler a luminosidade da imagem, o fotômetro divide-a em cinco partes e mede a luminosidade de cada uma. Um microprocessador compara as cinco leituras com valores já programados na câmara, valores es-

Microcomando

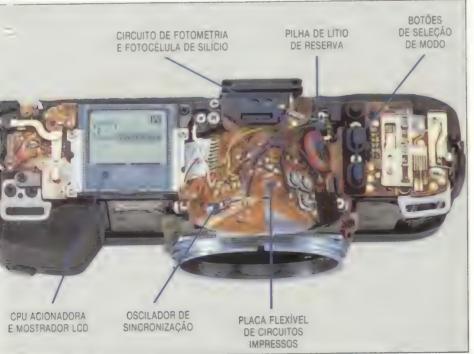
O microprocessador especial, de 8 bits, controla toda a operação da Canon T70 auxiliado por chips ISO e de fotometria. O oscilador gera os pulsos de sincronização e controla o tempo de exposição. Contatos de mola controlados pelo circuito integrado de fotomedição ajustam a abertura. Pode ser incorporado um módulo de comando opcional para possibilitar exposição automática fixada (entre um segundo e um dia), e que permite escrever os dados de sincronização diretamente no negativo.

CANON T70

PESO
715 g com objetiva de 50 mm.

OBJETIVA
50 mm x f1.8.

ABERTURA
f1.8 a f22

EXPOSIÇÃO
1/1.000 a 2 segundos.

ses resultantes da análise de milhares de fotos.

Mas, de todas as câmaras à venda, a que faz uso mais intenso da eletrônica é a Canon T70. Ela não tem controles mecânicos: todos os ajustes se fazem por meio de botões. Para selecionar um de seus oito modos, aperta-se um botão na parte superior esquerda da câmara; o modo escolhido surge indicado no mostrador de cristal líquido (LCD, Liquid Crystal Display) situado na parte superior. As informações mais importantes, como velocidade de obturação, abertura e modo de operação, aparecem também no próprio visor, de modo que o fotógrafo não precisa afastar a câmara do olho para os ajustes, enquanto enquadra o tema.

Funcionamento da Canon T70

Ao escolher uma velocidade de obturação, fazem-se os ajustes de acordo com o valor apontado no mostrador por intermédio de dois botões, para aumentar ou reduzir a velocidade. Procede-se de forma semelhante com a velocidade do filme, cuja sensibilidade vem indicada pelas unidades ASA ou ISO. O contador de avanço, que mostra o número de fotos batidas, também aparece no mostrador de cristal líquido.

A câmara tem um motor embutido (motor drive) para avançar o filme e rebobiná-lo quando acaba. A T70 usa duas pilhas comuns, cujas condições são apontadas no mostrador por três segmentos: a presença de todos indica carga plena; dois segmentos, meia carga; e apenas um, que as pilhas estão fracas, sendo necessário substituí-las. Quando se utiliza o disparador automático da câmara, o mostrador conta os segundos até que o obturador seja acionado.

Os microprocessadores usados na câmara são bem menos potentes que os dos computadores. A T70 dispõe de um processador de 8 bits, do tipo CMOS (Complementary Metal Oxide Semiconductor, "semicondutor complementar de óxido metálico"), que consome pouquíssima energia. Funciona numa freqüência de sincronização de apenas 32 kHz — os microcomputadores são cerca de cem vezes mais rápidos. Quando a câmara não está sendo usada, a freqüência se reduz a 8 kHz, para economizar energia.

Com ROM de grande capacidade, mas RAM de apenas 16 bytes, o microprocessador encontra-se instalado numa unidade plana de sessenta pinos. Quatro outros chips operam com ele. O mais importante, de entrada/saída, controla a operação mecânica da câmara. Também converte o sinal elétrico analógico do chip fotométrico em digital, permitindo sua interpretação pelo microprocessador.

A microeletrônica vem aumentando o prazer de fotografar com qualidade sem necessidade de entender as complexidades do processo. Sempre haverá casos, no entanto, de indicação inadequada da câmara, seja de exposição ou de foco. Assim, quem conhece seu funcionamento ainda leva vantagem sobre um principiante com equipamento sofisticado.

EFEITOS VISUAIS

A Commodore dedica muita atenção ao VIC-II, um chip controlador de vídeo. Analisaremos aqui, também, o funcionamento de seus registradores na produção de vários efeitos, assim como a organização de sua memória.

Existem basicamente oito modos gráficos no Commodore 64. Usando baixa resolução, a fonte de caracteres pode estar tanto em RAM como em ROM e ser apresentada em três modos: normal, multicor e multicor expandido. Em alta resolução, usam-se dois modos: normal e multicor.

Mas outras variações são possíveis. Por exemplo: usar uma tela com 38 colunas, e não com 40, como seria normal, e/ou 24 linhas em vez de 25. Em geral, utilizam-se esses recursos em conjunto com o rolamento da tela (scroll) na horizontal ou na vertical, para produzir efeito de movimento numa animação.

Se a tela de alta resolução é usada em conjunto com um programa longo, a memória disponível escasseia. Mas o Commodore 64 permite considerável liberdade na localização das telas de baixa ou de alta resolução na memória.

A função do chip de interface de vídeo 6566/67 (VIC-II) consiste em gerar, para apresentação em tela, os dados enviados ao televisor ou monitor. Para isso é necessário que o VIC-II seja capaz de ler a tela, em forma de dados, tanto da RAM como da ROM.

Modos gráficos

No modo normal, só se obtêm duas cores: "cor" e "não cor" (aparecendo, nesse caso, a cor existente no fundo). Já o modo multicor — quatro cores possíveis numa única célula de caractere — pode ser usado tanto em baixa como em alta resolução, embora varie a determinação da cor utilizada. Estes comandos ligam e desligam o modo multicor:

```
POKE 53270,PEEK(53270)OR 16
POKE 53270,PEEK(53270)AND 239
```

Se o quarto bit do nybble (meio byte) de cor associado corresponde a 1, o caractere é interpretado no modo multicor, com os 3 bits mais baixos determinando a cor. Significa que os caracteres que têm o nybble de cor associado no intervalo entre 0 e 7 recebem interpretação normal, enquanto os que se encontram entre 8 e 15 são interpretados como multicor. Determinam-se as cores de cada par de pixel conforme a tabela 1.

Mudando o conteúdo dos endereços 53282 e 53283, alteramos instantaneamente as cores de cada par de pixel multicor associado. Esse modo trabalha melhor com caracteres definidos pelo usuário, ou seja, convém escolher novo padrão de bits levando em conta que serão interpretados aos pares.

O modo multicor expandido permite controle sobre a cor de fundo dos 64 primeiros caracteres da matriz, mas não pode ser usado em conjunto com o modo multicor. As seguintes instruções em BASIC ligam e desligam o modo multicor expandido:

```
POKE 53265,PEEK(53265)OR 64
POKE 53265,PEEK(53265)AND 191
```

O mesmo caractere aparece na tela com até quatro cores de fundo. O restante dos caracteres não pode ser usado nesse modo, pois utilizam-se os bits 6 e 7 do código de tela para controlar indiretamente a cor do caractere. A tabela 2 mostra como os códigos de tela e os registradores de cor estão relacionados.

Gerenciamento de memória

O chip VIC-II "vê" a memória de forma diferente e muito mais simples que o 6510. Em qualquer momento, considera apenas um dos quatro blocos — ou bancos — de 16 Kbytes de memória. Podemos imaginar esse banco de 16 Kbytes como uma "janela" do VIC-II para a memória.

O endereço-base dessa janela pode assumir um dos quatro valores sob controle de software:

```
1000 REM ** SELECIONA BANCO PARA
     JANELA VIC **
1010 POKE 56578,PEEK(56578)OR 3
1020 JN=3: REM * SELECIONA JANELA
     NORMAL *
1030 POKE 56576,(PEEK(56576)AND 252)OR JN
```

Tabela 1

Padrão de bits do par de pixels		Cor	Determinado por
0	0	Cor de fundo	53281 ($D021) Bits 0 a 4
0	1	Multicor 1	53282 ($D022) Bits 0 a 4
1	0	Multicor 2	53283 ($D023) Bits 0 a 4
1	1	Cor de linha	Bits 0 a 3 do nybble de cor

Tabela 2

Código de tela	Bit 7	Bit 6	Registrador da cor de fundo
64 - 127	0	1	53282 ($D022) Cor expandida 1
128 - 191	1	0	53283 ($D023) Cor expandida 2
192 - 255	1	1	53284 ($D024) Cor expandida 3

Aqui, JN tem valores de 0 a 3. Em qualquer momento, obtém-se o valor atual de JN com PEEK (56576)AND3. Podemos calcular o endereço do início de cada janela a partir da fórmula JI = 16384 * (3-JN). Dela extraímos a seguinte tabela:

JN	Endereço do início da janela
0	49152 ($C000)
1	32768 ($8000)
2	16384 ($4000)
3	00000 ($0000)

Dentro de uma janela do VIC-II, o 6510 espera encontrar a memória de tela e a imagem da ROM de caracteres, quando usada a baixa resolução. Eventualmente, precisa de dados de sprites e, se for o caso, pesquisa-os dentro da mesma janela. Os indicadores de cada um dos oito sprites são colocados no final da memória de tela (portanto, é necessário movê-los juntamente com a janela).

A memória de tela ocupa 1.024 bytes de memória e pode ser colocada em dezesseis diferentes lugares dentro de uma janela (1.024 * 16 = 16.384, ou seja, 16 Kbytes). Para tanto, usam-se os 4 bits mais altos do registrador encontrado no endereço 53272 ($D018).

```
1040 REM ** SELECIONA POSICAO DA TELA **
1050 TE = 1 : REM SELECIONA POSICAO
        NORMAL **
1060 POKE 53272,(PEEK(53272)AND15)OR
        16 * TE
```

Aqui, TE pode assumir valores de 0 a 15. Em qualquer momento, PEEK(53272)AND 15 fornece o valor atual de TE. O endereço do início da tela correspondente é calculado por SC = JI + 1024 * TE.

Um problema na mudança de posição da tela na memória decorre de a RAM de cor não mudar. Portanto, é necessário usar uma pequena sub-rotina em linguagem de máquina para transferi-la para um buffer, e vice-versa. Outro fator a considerar: quando se utiliza um comando PRINT, torna-se necessário informar ao sistema operacional onde se encontra a memória de tela. Resolve-se isso com um POKE (ou STA) no endereço 648 ($0288). Esse indicador contém o maior byte do endereço do início da tela. Se SC foi devidamente computado, este comando BASIC fará o trabalho:

```
POKE 648,INT(SC/256)
```

Para selecionar o endereço inicial da matriz de caracteres ou da tela de alta resolução, usa-se o seguinte conjunto de instruções:

```
1070 REM ** SELECIONA ENDERECO INICIAL
        DA TELA DE ALTA RES. OU MATRIZ DE
        CAR.
1085 HO = 4 : REM ** POSICAO NORMAL
1090 POKE 53272,(PEEK(53272)AND240)OR 2 * HO
```

Em princípio, HO pode ter qualquer valor entre 0 e 7, mas, na prática, outros fatores limitam as opções. Com JN equivalendo a 1 ou 3, HO não tem valores 2 ou 3, pois é aí que o VIC-II vê a imagem normal da ROM de caracteres. A fórmula CM = JI + 2048 * HO nos dá o endereço que se encontra ativo no momento.

Ajustando esses registradores, movemos a memória de tela para qualquer lugar na RAM segundo requeiram nossos programas.

Dividindo o tempo com a CPU

Para produzir uma imagem de vídeo, o VIC-II precisa ler os dados que gerarão essa tela na memória. Permite-se, portanto, que o VIC-II tenha acesso a algumas linhas do bus de endereçamento e a todas as linhas do bus de dados. O processo pelo qual o VIC-II utiliza a ROM ou a RAM em conjunto com o 6510 é o acesso direto à memória (DMA, Direct Memory Access).

O ideal seria que o VIC-II só usasse as linhas de endereço e de dados quando o 6510 não as estivesse utilizando, de forma que a ação de um processador não interferisse na do outro. Mas não é o caso e, como o VIC-II deve ler grande número de dados — sobretudo quando lida com vários sprites simultaneamente —, não há tempo suficiente para operar de modo independente. Assim, o VIC-II dispõe de uma linha de controle especial, a linha de bus disponível (BA, Bus Available), com a qual pode emitir um sinal de espera (hold off) ao 6510. Isso fornece a necessária reserva de tempo para que ele possa trabalhar.

A primeira interrupção de varredura muda para o modo de alta resolução

A segunda interrupção de varredura retorna ao modo texto

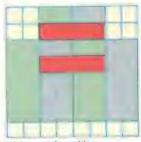
Os dois lados da história

Muitos jogos comerciais de aventura desenvolvidos para o Commodore 64 empregam técnicas de interrupção de varredura para produzir, simultaneamente, texto e imagens de alta resolução. A tela mostrada acima é do Spiderman, um jogo criado por Scott Adams. Nele, o terço superior da tela apresenta a imagem de um local do jogo, enquanto os dois terços restantes são utilizados pelo texto que descreve a cena. O dispositivo de varredura do vídeo é programado para gerar duas interrupções: uma no topo da tela e outra um terço mais abaixo. Toda vez que ocorre uma interrupção de varredura, uma rotina apropriada alterna entre o modo texto e o de alta resolução.

Cores coordenadas

No modo de apresentação gráfica normal do Commodore 64, cada bit 1 do conjunto de 8 bytes que definem um caractere é exibido na cor de linha vigente. Os bits 0 surgem na cor escolhida para o fundo da tela. No modo multicor, os bits são interpretados aos pares. As quatro combinações possíveis de um par de bits representam as cores de linha e de fundo e duas multicores adicionais.

"A" no modo normal

```
0 0 0 1 1 0 0 0
0 0 1 1 1 1 0 0
0 1 1 0 0 1 1 0
0 1 1 1 1 1 1 0
0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0
0 0 0 0 0 0 0 0
```

"A" no modo multicor

```
00  01  10  00
00  11  11  00
01  10  01  10
01  11  11  10
01  10  01  10
01  10  01  10
01  10  01  10
00  00  00  00
```

Chave

01 Multicor 1

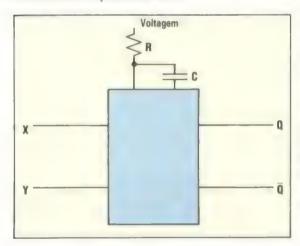
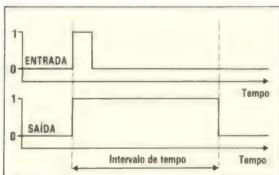
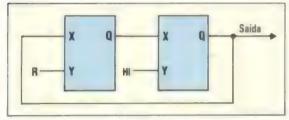
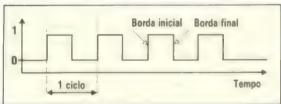
10 Multicor 2

11 Cor de linha

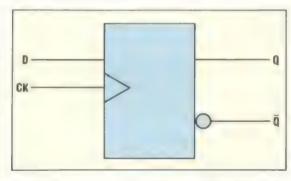
00 Cor de fundo

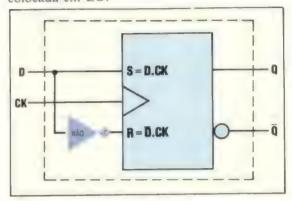
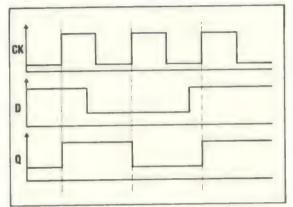
CIRCUITOS DE SINCRONIZAÇÃO

O controle das funções internas do computador exige cronometragem precisa. Examinam-se aqui três tipos de circuitos que produzem os sinais de sincronização necessários a tal trabalho.

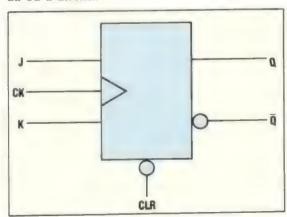
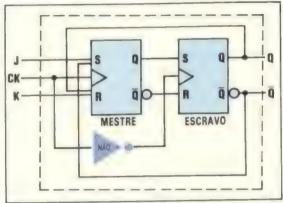
A principal utilidade do circuito monoestável é tornar possível a introdução de intervalos de tempo fixo nas operações dos circuitos lógicos. Quando um circuito desse tipo recebe a entrada de um pulso, a saída é colocada em 1 (HI, "alto") por um intervalo de tempo fixo, antes de retornar a seu estado normal de saída 0 (LO, "baixo"). O período em que a saída permanece HI é determinado pelos valores de certos componentes do circuito monoestável, o qual pode ser visto no esquema abaixo:

Aciona-se esse dispositivo modificando-se X de HI para LO, ou Y, de LO para HI. O tempo da saída depende dos valores de R e C. O gráfico a seguir mostra como se relacionam a entrada e a saída:

A duração da saída HI permite, por exemplo, controlar o motor gradual de uma leitora de fita ou retardar a transmissão de um bit. Podem-se vincular dois circuitos monoestáveis para produzir um pulso de clock, que oscila a intervalos fixos entre HI e LO:

A saída produzida assemelha-se a uma onda quadrada (como mostram os gráficos). O intervalo de tempo entre dois instantes de HI (ou de LO) sucessivos constitui um ciclo, que, em geral, dura 1 milionésimo de segundo. Esse sinal repetido do clock é a batida do coração do microcomputador, ordenando as muitas funções realizadas na CPU. O diagrama seguinte indica os nomes atribuídos às "bordas" do gráfico da onda quadrada, onde um pulso passa de HI para LO ou vice-versa.

Vamos examinar dois tipos de flip-flop cuja atividade é controlada pelos pulsos regulares do clock.

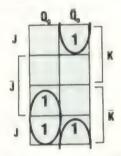
Flip-flop tipo D

Este circuito possui uma entrada lógica (D) e uma entrada de clock (CK):

O desenho do tipo D baseia-se no flip-flop R-S, examinado anteriormente. No entanto, é a entrada do clock que produz o método especial de operação conhecido como travamento. A saída do circuito, Q, fica determinada no início de um ciclo do clock. Se, nesse momento, a entrada em D for HI, a saída Q será colocada em HI. Mas,

se a entrada em D estiver em LO, a saída Q será colocada em LO.

Esses gráficos mostram que a saída Q se modifica somente durante a transição no clock de LO para HI. Daí, o tipo D ser chamado também de flip-flop de "borda ascendente ativada".

Flip-Flop J-K

Esse tipo de circuito (ilustrado abaixo) é também denominado mestre-escravo, por incluir dois flip-flops R-S numa relação dominante-dominado. No mesmo ciclo do clock, o dispositivo armazena o pulso de entrada num dos flip-flops, enquanto fornece saída proveniente da outra unidade dependente da entrada anterior. Temos um exemplo disso na operação de deslocamento comum à maioria dos processadores: deslocam-se bits dentro dos registros uma posição à esquerda ou à direita.

O diagrama a seguir mostra como se unem os dois tipos R-S — o mestre e o escravo. Suponhamos que se aplicou uma entrada em J ou K: ela será fornecida ao mestre se o pulso do clock estiver HI, e ao escravo se a entrada do clock for LO, já que os tipos R-S são bordas descendentes ativadas. Assim, em cada momento, apenas o tipo R-S é acionado, ficando a entrada anterior "alojada" dentro da outra:

A tabela de estado para o flip-flop J-K *(ao lado)*, assemelha-se às tabelas de validação, mas utiliza uma variável Q_0 — a saída anterior. As entradas HI simultâneas em J e em K fazem com que o flip-flop mude de estado a cada pulso do clock. Conhecida como articulação, essa mudança ocorre em função do fornecimento de entradas escravas para as entradas mestras. Com um flip-flop R-S, essa é uma combinação de entrada desativada, com saída indefinida. Da tabela, obtém-se o mapa-k, abaixo, considerando Q_0, J e K como entradas.

Q_0	J	K	Q
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Desse mapa-k resulta a expressão lógica

$$Q = Q_0 . J + Q_0 . K$$

Essa é a equação característica do flip-flop J-K.

Respostas do exercício anterior

1) Um flip-flop é também conhecido como biestável: fica estável quando Q = 1 e Q = 0 ou quando Q = 0 e Q = 1.

2) a) Não é um estado estável.
b) Considerando primeiro a entrada superior, o flip-flop passará para RESET e para SET, se for a porta inferior.
c) Sim (veja resposta anterior).
d) Para que todos os registros atinjam um estado previsível quando se liga a energia, eles devem ser estabelecidos ou restabelecidos na inicialização do sistema do computador.

MEMÓRIA PAGINADA

Saber como funciona a memória dos computadores é fundamental para a programação em ASSEMBLY. Vamos examinar as restrições à paginação da memória e à operação da CPU impostas pelo uso do sistema binário.

No artigo anterior explicamos com uma analogia o modo como um computador armazena informações sob a forma de corrente elétrica. Usamos o exemplo da fábrica em que cada operário possui uma combinação individual de interruptores para acender quatro lâmpadas na gerência, identificando-se assim quem está trabalhando. Isso mostrou como é possível representar a informação (ou seja, o nome da pessoa que está trabalhando) por meio de um fluxo de eletricidade.

Em nosso exemplo, descobrimos que, usando quatro interruptores e lâmpadas, podíamos representar os números de 0 a 15. Em outras palavras, havia apenas dezesseis combinações possíveis. Entretanto, se, em vez disso, tivéssemos usado oito interruptores e lâmpadas, teríamos obtido 256 combinações diferentes ($2\times2\times2\times2\times2\times2\times2\times2=256$) e, portanto, seríamos capazes de contar de 0 a 255.

Em seu microcomputador, a memória está distribuída em grupos de oito interruptores, cada um denominado byte. Em geral, a CPU manipula informações a 1 byte por vez, o que significa que ela pode apenas somar, comparar e armazenar números entre 0 e 255. Pode parecer que isso limita sua capacidade aritmética, mas não é o caso. Se você pensar numa adição como, por exemplo, 63.951 + 48.770 = ?, reconhecerá que, na prática, manipula os dígitos individuais um de cada vez. Do mesmo modo, a CPU efetua operações aritméticas com números grandes usando um byte de cada vez.

Por conter oito interruptores, cada byte armazena um número binário de oito dígitos. Cada uma dessas posições de dígito binário chama-se bit — a menor unidade de informação. Num byte, um bit sempre equivale a ligado ou desligado; um dígito binário é sempre igual a 1 ou 0.

Muitas vezes, é importante considerar um determinado bit num byte, de forma que se convencionou numerar os bits de 0 a 7, da direita para a esquerda. Se um byte contém o número binário 00000001, dizemos então que o bit 0 é 1, ou que o bit 0 está "ligado" ou, ainda, está "set"; todos os outros bits são 0, ou seja, estão "desligados", ou "clear". Assim, no número binário 01001000 temos: os bits 3 e 6 estão set, o bit 4 está desligado, o bit 7 é 0, o bit 0 está clear, e etc. Num byte, o bit 0 também se chama bit menos significativo (LSB, Least Significant Bit), e o 7, bit mais significativo (MSB, Most Significant Bit).

Podemos, então, conceber a memória do computador como uma longa tira de papel quadriculado, com oito quadrados de largura e milhares de comprimento. Cada linha de oito quadrados seria um byte e cada quadrado representaria um bit.

A memória será inútil se você não puder localizar cada uma de suas posições. Assim, cada byte possui um rótulo de identificação, chamado endereço, que não se encontra escrito em parte alguma do papel (ou do byte); ele é simplesmente o número do byte na memória, contando-se a partir do início desta. O primeiro byte tem, portanto, endereço 0, o seguinte tem endereço 1, o próximo, endereço 2, e assim por diante. Se você quiser gravar algo no byte 43, começará na parte inferior da memória (no 0) e contará, byte após byte, até chegar ao 43. Nada, porém, o identificará como byte 43, exceto sua posição.

Esse sistema de endereçamento de memória seria ótimo se houvesse apenas umas poucas centenas de bytes. A CPU pode contar de 0 a 100 em frações de milissegundo, mas os computadores possuem milhares de bytes, e contar de 0 a 20.000 leva um tempo razoável, mesmo para um microprocessador. O computador supera esse problema dividindo a memória em páginas, assim como num livro.

Vamos continuar imaginando a memória do computador como uma tira de papel com milhares de quadrados de comprimento por oito de largura. Podemos pensar no corte dessa tira a cada 100 bytes (isto é, cortamos entre o byte 99 e o 100; entre o 199 e o 200; entre o 299 e o 300; etc.). Cada uma dessas tiras de papel fica sendo então uma página de 100 bytes. A página 0 começa no byte 0 e continua até o byte 99; a página 1 começa no byte 100 e vai até o 199; a página 2 vai do byte 200 ao 299; etc. Agora, para encontrar qualquer byte, digamos o 3518, não precisamos contar 3.518 bytes desde o início da memória, pois sabemos, pelo endereço, que esse byte está na página 35. Só precisamos contar 35 páginas pela parte inferior da memória e, então, contar os bytes, a partir do pé da página, até alcançarmos o de número 18.

Esse sistema de memória paginada é conveniente porque podemos olhar para o endereço de qualquer byte e dividi-lo em duas partes: os dígitos da coluna das centenas para a esquerda representam o número de página do byte; e os dígitos da coluna das dezenas para a direita são o número do byte, contado a partir do início da página. No exemplo acima, dividimos de fato o endereço 3518 em dois números: número de página 35 e número de byte 18, nessa página. Chamamos 18 de offset (deslocamento), ou offset de página, porque é o número que dá o quanto você deve deslocar (ou aumentar), numa página, o endereço de seu byte inicial, a fim de atingir o byte procurado.

O computador, no entanto, não conta em decimais, como nós fazemos — ele conta em binários. O sistema de paginação depende da capacidade de encontrar a página e o offset simplesmente inspecionando o endereço do byte. O endereço decimal 99 é representado por 01100011 em binário; e o decimal 100 por 01100100; o decimal 199 é 11000111 em binário; e 200 é 11001000. Podemos ver, nesses exemplos, que não há um modo simples de usar os números binários para discriminar, um após o outro, os números de página, como fazemos facilmente com os decimais. A razão disso é a escolha do tamanho da página.

Escolhemos 100 como o tamanho da página precisamente por ser um número significativo no sistema decimal (uma potência de 10). Se, no entanto, temos de contar em binário, devemos escolher outro tamanho de página. O tamanho de página utilizado pelos computadores é 256, de modo que a página 0 começa no byte 0 e vai até o 255; a página 1 começa no byte 256 e vai até o 511; e assim por diante. Para reconhecer a conveniência disso, devemos escrever esses endereços em binário:

Página 0: do byte 00000000 até o byte 11111111
Página 1: do byte 100000000 até o byte 111111111

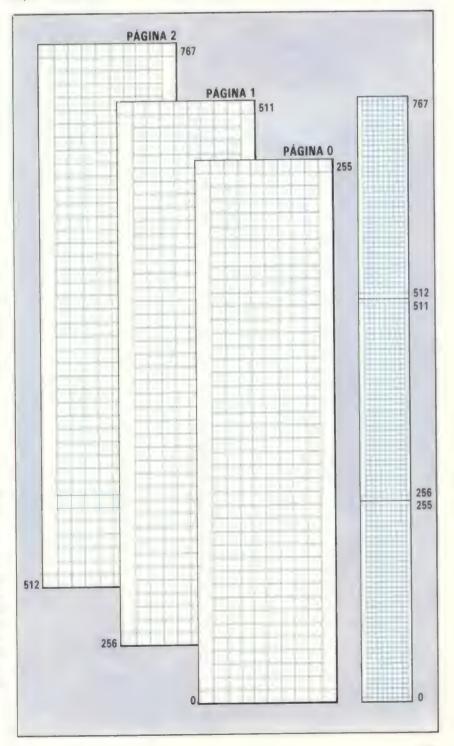
Como se pode ver, podemos contar em binário, de 0 a 255, com números de 8 bits; o número seguinte — 256 — requer 9 bits, e com 9 bits podemos contar até 511; etc. Assim, se o tamanho da página é 256, e se contamos em binário, então o offset são os 8 bits mais à direita, sendo o número da página dado pelos bits da esquerda.

Isso pode parecer complicado, visto termos estabelecido que a CPU só pode manipular bytes isolados, e 1 byte contém apenas 8 bits. Você poderia perguntar, então, de que adianta falar em números de 9 e 10 bits? A resposta é que todos os endereços na memória são tratados como números de 2 bytes, e a CPU lida com um deles de cada vez. Se reescrevermos os limites da página como números de 2 bytes, o sistema ficará mais claro:

Página 0: começa em 00000000 00000000
termina em 00000000 11111111
Página 1: começa em 00000001 00000000
termina em 00000001 11111111
Página 2: começa em 00000010 00000000
termina em 00000010 11111111
Página 3: começa em 00000011 00000000
termina em 00000011 11111111

e assim por diante.

Agora podemos ver que, quando a CPU "pesca" ou coloca informação num byte da memória, esse byte é identificado por um endereço de 2 bytes. O primeiro, que está à esquerda, dá o número da página, enquanto o segundo, à direita, dá o offset.

Endereçamento paginado
O endereçamento paginado divide a memória em blocos imaginários, ou páginas, de 256 bytes. Todos os endereços são expressos como números de 2 bytes: um fornece o número da página e o outro dá o offset relativo ao início desta.

Os números hexadecimais constituem um outro tipo de representação, bastante diferente. Ao invés de apenas dígitos de 0 a 9, como no sistema decimal, são usados dígitos de 0 a 9 e letras de A a F, correspondendo as letras aos seguintes valores:

A = 10; B = 11; C = 12; D = 13; E = 14; F = 15;

Pode parecer estranho que se queira utilizar esse sistema de numeração quando se trabalha com computadores, mas o fato de o número 16 ser uma potência do número 2 — isto é, $16 = 2 \times 2 \times 2 \times 2$ — traz muitas vantagens.

Com apenas um dígito hexadecimal, representa-se um nybble, ou seja, quatro dígitos binários. Veja a tabela a seguir:

DÍGITO HEXADECIMAL		DÍGITOS BINÁRIOS
0	=	0000
1	=	0001
2	=	0010
3	=	0011
4	=	0100
5	=	0101
6	=	0110
7	=	0111
8	=	1000
9	=	1001
A	=	1010
B	=	1011
C	=	1100
D	=	1101
E	=	1110
F	=	1111

A grande vantagem de se trabalhar com o sistema hexadecimal está no fato de cada página poder ter seu número representado apenas por dois dígitos, bem como o offset. Desse modo, o endereço de qualquer posição na memória é dado por quatro dígitos, e seu conteúdo por apenas dois.

Veremos agora um programa — para o TK 85 e o TK 90X — que apresenta um endereço e seu conteúdo de várias maneiras:

Endereço — em binário, em hexadecimal e em decimal.

Conteúdo — em binário, em hexadecimal, em decimal e em código ASCII.

Este programa roda no TK 85 e também no TK 90X

```
  10 REM       ESPIAO
  20 REM
  30 REM PARA A LINHA SINCLAIR
  40 REM
  50 LET A$="0123456789ABCDEF"
  60 LET B$="0000 0001 0010 0011
 0100 0101 0110 0111 1000 1001 1
010 1011 1100 1101 1110 1111"
  70 DIM C$(4)
  80 DIM D$(2)
  90 LET A=10
 100 LET B=11
 110 LET C=12
 120 LET D=13
 130 LET E=14
 140 LET F=15
 150 CLS
 160 PRINT "ENTRE O ENDERECO INI
CIAL"
 170 INPUT G
 180 PRINT "ENTRE QUANTOS BYTES"
 190 INPUT H
 200 CLS
 210 FOR X=G TO (G+H)
 220 PRINT X,"█";PEEK X
 230 LET C$(1)=A$(1+INT (X/4096
)
 240 LET I=X-4096*INT (X/4096)
 250 LET C$(2)=A$(1+INT (I/256))
 260 LET I=I-256*INT (I/256)
 270 LET C$(3)=A$(1+INT (I/16))
 280 LET I=I-16*INT (I/16)
 290 LET C$(4)=A$(1+I)
 300 LET D$=A$(1+INT ((PEEK X)/1
6))+A$(1+(PEEK X)-16*INT ((PEEK
X)/16))
 310 PRINT C$,"█",D$
 320 FOR Y=1 TO 4
 330 PRINT B$((VAL C$(Y)+1)*5-4
TO (VAL C$(Y)+1)*5-1);
 340 NEXT Y
 350 PRINT "█",
 360 PRINT B$((VAL D$(1)+1)*5-4
TO (VAL D$(1)+1)*5-1),
 370 PRINT B$((VAL D$(2)+1)*5-4
TO (VAL D$(2)+1)*5-1),
 380 PRINT
 390 PRINT "████████████████████
████████"
 400 NEXT X
```

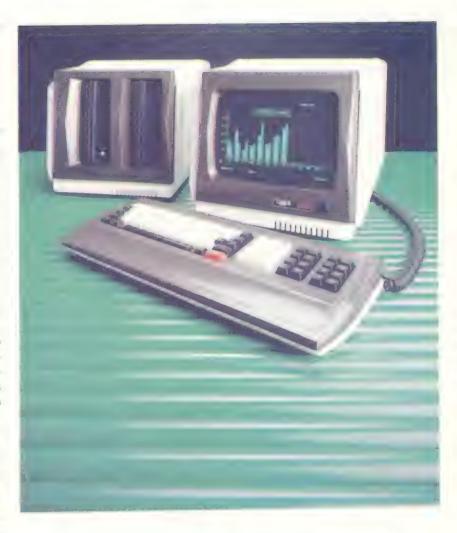
COBRA

Tirando partido da reserva de mercado e da associação com os bancos, a Cobra consolidou-se no mercado nacional, numa história que se confunde com a da política da informática brasileira.

Em fins da década de 60, um grupo de oficiais da marinha brasileira percebeu a necessidade de auto-suficiência na produção de equipamentos de eletrônica digital. E não foi preciso muito tempo para que o projeto de criação de uma indústria brasileira de computadores tomasse corpo: no início dos anos 70, por ocasião de uma compra à Inglaterra de embarcações de combate com artilharia eletrônica, o almirantado espantou-se com o preço dos computadores ingleses. Era tão elevado que a marinha optou por adquirir apenas os navios e por desenvolver aqui mesmo o hardware e o software necessários. Nascia assim, em julho de 1974, com o objetivo de projetar, produzir e comercializar equipamentos para a área de computação, a Cobra - Computadores e Sistemas Brasileiros S.A., empresa cuja história se liga estreitamente à da política da informática nacional.

A iniciativa do governo estabelecia como prioridade a absorção da tecnologia de ponta estrangeira para posterior desenvolvimento no país, de modo que, em alguns anos, a indústria nacional atingisse as condições para uma vida independente. A idéia era criar empresas nacionais associadas a estrangeiras, e, seguindo essa premissa, em sua formação societária inicial, a Cobra apresentava a participação da Digibrás - Empresa Digital Brasileira S.A., da E. E. Equipamentos Eletrônicos S.A., e da tradicional firma inglesa Ferranti Ltd.

Um contrato de licenciamento entre a Cobra e a Ferranti garantiu o fornecimento da tecnologia necessária à fabricação dos computadores e sistemas da família Argus 700. Ao mesmo tempo que enviava técnicos ao exterior para treinamento, o governo brasileiro tomou certas medidas no sentido de estimular o consumo do produto nacional. E as principais decisões nessa área foram o controle das importações e o impedimento de produção e comercialização de equipamentos que não implicassem transferência de tecnologia ao Brasil. Essa decisão não transcorreu sem traumas, decorrentes principalmente de conflitos com as multinacionais do setor. Afinal, na lista de importações, os computadores ocupavam o terceiro lugar.

Micro profissional
O Cobra 210 foi planejado para uso em pequenas e médias empresas, e instituições de pesquisa científica. Ele faz parte de uma família que também inclui dois tipos de terminais inteligentes: o TI 200 (assíncrono) e o TR 207 (síncrono).

Enfrentando a concorrência

Percebendo a impossibilidade de competir com as gigantes estrangeiras na produção de equipamentos de grande porte, a indústria nacional procurava um espaço que permitisse seu desenvolvimento e auto-suficiência. Assim, em 1976, a Capre (Coordenadoria das Atividades de Processamento Eletrônico), órgão governamental substituído em 1979 pela SEI (Secretaria Especial de Informática), definia as primeiras linhas da política de reserva de mercado: por um período de sete anos, a industrialização no setor de mínis e micros estaria aberta apenas às empresas nacionais. (A luta pela reserva de mercado estendeu-se até outubro de 1984, quando o Congresso aprovou a Lei de Informática.)

A escolha pelo setor de mínis e micros prendia-se a uma razão muito forte. Ao contrário dos grandes computadores, o componente eletrôni-

Centro de tecnologia
Unidade industrial da Cobra, com 25.000 m². localizada em Jacarepaguá, no Rio de Janeiro.

co principal desses equipamentos eram os chips, facilmente comprados no exterior, e aos quais os fabricantes das máquinas maiores não pareciam dar atenção.

Ainda em 1976, já transformada em sociedade anônima, a Cobra firmou um contrato de cooperação técnico-industrial com a Sycor Inc., que a capacitou a dar início à fabricação dos equipamentos da linha 400. Para dar suporte adequado aos usuários brasileiros, uma equipe de trinta engenheiros e técnicos da empresa recebeu treinamento nos Estados Unidos.

O grande salto da empresa, no entanto, ocorreu em 1977. A Cobra foi autorizada pelo governo a aumentar seu capital social, o que resultou em reestruturação e numa redefinição de sua linha de produtos. Contando na composição acionária com capitais privados, principalmente de bancos e empresas estatais, a Cobra tornou-se uma empresa de economia mista.

O papel dos bancos

A entrada dos bancos nesse mercado foi o principal impulsionador para o grande salto da indústria nacional. Com as importações sob severo controle e a crescente necessidade de processamento de dados em seus serviços, as organizações bancárias viam com simpatia a possibilidade de se associarem à empresa — sócias e clientes. Mas a participação dos bancos, com representantes dentro do Conselho de Administração, embora apresente vantagens, cria uma situação insólita. Entre os acionistas encontram-se, por exemplo, o Bradesco e o Itaú, que possuem duas das maiores empresas concorrentes da Cobra. Fica difícil, então, aos diretores da Cobra levarem seus planos estratégicos ao Conselho de Administração.

Os contratos com as empresas estrangeiras Ferranti e Sycor garantiram a transferência e absorção de tecnologia, resultando nos primeiros produtos com características próprias, que atendiam às necessidades do mercado nacional: o Cobra 700 II, o Cobra 400 II e o Cobra 400 M, mais potentes que os similares estrangeiros.

Consolidação

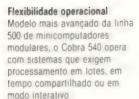
Em apenas três anos, a empresa brasileira absorveu completamente as tecnologias inglesa e americana. Já em 1977, um contrato de transferência de tecnologia assinado com o Serpro (Serviço Federal de Processamento de Dados), um dos acionistas da Cobra, permitiu à empresa dispor de um corpo técnico adequado à produção de equipamentos inteiramente nacionalizados.

Em 1980, com seis anos de existência e estabelecida definitivamente em Jacarepaguá, no Rio de Janeiro, a Cobra possuía significativa linha de produtos — um microcomputador para aplicações comerciais de grande porte (o Cobra 300), e quatro linhas de terminais: TR, para comunicação à distância; TD, para entrada de dados; TF, para aplicações financeiras; e TI, assíncronos inteligentes.

No mesmo ano, a empresa alcançou seu objetivo inicial com o lançamento do primeiro computador totalmente projetado, desenvolvido e fabricado no Brasil, o Cobra 530. Consolidava-se, portanto, uma indústria brasileira com capacitação tecnológica equivalente às existentes no mercado internacional.

Em julho de 1984, três novos produtos foram introduzidos no mercado: o microcomputador Cobra 210 e os terminais TI 200 e TR 207. E, cinco meses mais tarde, um novo microprocessador veio acrescentar-se à linha de produtos da empresa: o Cobra 480, de 16 bits, compatível com a família 500, que inclui os mínis 520, 530 e 540.

A composição acionária da Cobra está assim distribuída: 62,4% do capital encontra-se em poder da União, através do BNDES, Caixa Econômica Federal, Banco do Brasil, Serpro, Digibrás e Eletrônica Digital do Brasil. O restante se divide entre companhias privadas nacionais e estrangeiras.

Flexibilidade operacional
Modelo mais avançado da linha 500 de minicomputadores modulares, o Cobra 540 opera com sistemas que exigem processamento em lotes, em tempo compartilhado ou em modo interativo

MÚSICA EM SOFTWARE

A MIDI, entre todos os sistemas disponíveis, é o que oferece aos músicos a mais ampla gama de aplicações, permitindo melhorar o desempenho tanto em execuções ao vivo como em estúdio.

A MIDI (Musical Instrument Digital Interface, "interface digital para instrumentos musicais") oferece aos estudantes vantagens inquestionáveis. Aprender a ler música costuma ser difícil, mesmo quando o iniciante já é músico competente. Os primeiros estágios, quando se fazem exercícios com sustenidos, bemóis e sinais de marcação de tempo, podem ser demorados e trabalhosos, e a correspondência entre os sinais no papel e o som que representam nem sempre é óbvia.

Grande parte do problema se deve à própria natureza da música: a harmonia depende da duração das notas e dos intervalos. Assim, se o iniciante precisa interromper a melodia para acompanhar a notação visual, todos os indicadores de intervalos de tempo na partitura perdem o sentido. De modo semelhante, ele pode levar algum tempo tentando interpretar determinada seqüência ou compasso; enquanto isso está sendo feito, a música terá avançado.

A MIDI elimina esses obstáculos, permitindo que uma execução em sintetizador seja armazenada na memória do computador. Com o software apropriado, obtém-se na tela do micro a representação gráfica da melodia que está sendo tocada. Trata-se de um auxílio inestimável para qualquer estudante de música: se um dó central (do piano) é tocado no sintetizador, a tela mostra um pentagrama com essa nota; tocando um acorde em si menor, seus componentes harmônicos — si, ré e fá — aparecem no monitor juntamente com a duração.

Essa idéia pode ser ampliada por um software que toca uma peça pré-programada no sintetizador interfaceado, enquanto a correspondente partitura completa vai aparecendo na tela. Nessa situação é possível interromper simultaneamente tanto a música quanto sua notação e reiniciá-las a partir de determinado compasso. Além disso, pode-se variar o som da música, como um todo, mudando os parâmetros de controle no sintetizador — introduzindo, assim, o usuário na arte do arranjo.

Atingindo certo grau de confiança na leitura da música, ficará mais fácil escrevê-la com auxílio do teclado alfanumérico de um computador. Isso pode envolver a introdução de dados de execução sem qualquer referência imediata a um som do sintetizador e a confrontação do resultado com aquilo que se pretendia. Uma vez adquirida essa habilidade, pode-se substituir a notação no pentagrama convencional em favor de outro sistema, como a MCL (Music Composition Language, "linguagem de composição musical").

Para a música eletrônica, a MCL revela-se o meio mais apropriado de introdução de dados,

Novos sons, novo estilo
Entre os sintetizadores da série DX da Yamaha, o DX7 incorpora um método novo de fabricar o som, e de síntese da FM, antes restrito a aparelhos de custo muito elevado. Em vez de partir de um som já existente e modificá-lo depois, passando-o por filtros ou ajustando controles de envoltória, o DX7 cria seus próprios sons complexos combinando de várias maneiras seus "operators" (memórias geradoras senoidais). Como resultado, o som obtido com o DX7 aproxima-se daquele dos instrumentos acústicos (vibrafone, cravo, baixo) de maneira bastante convincente. O DX7 também incorpora controle de respiração, de forma que um músico pode soprar num receptor e acrescentar aos sons de um saxofone ou de um trompete, por exemplo, variações que têm a respiração como fonte. O DX7 pode utilizar programas em ROM com características de som gravado, ou pode armazenar, em programas em RAM, sons criados pelo músico. O DX7 é vendido por um preço em torno de 2.000 dólares.

Som sintetizado

Existem vários sintetizadores ativados a partir das cordas de uma guitarra, em vez de um piano. O Roland GR700 por exemplo. Informações sonoras complexas são recebidas por um captador hexafônico (seis sons) e enviadas ao sintetizador, onde se acrescentam parâmetros que não provêm da guitarra. O SynthAxe, sintetizador controlado por um microprocessador 6809, emprega a mesma técnica, captando dados sonoros de um contato elétrico entre a corda e o traste. Sensores na parte superior da cravelha apanham variações de bending e de deslizamento. Um segundo conjunto de cordas e um pequeno teclado processam ainda mais o som.

uma vez que inclui a especificação de características exclusivas da produção eletrônica do som. Nenhum sistema padronizado foi desenvolvido ainda para a aplicação da MCL — as máquinas têm suas próprias MCLs. A notação em pauta, embora seja um guia visual imediato muito útil, é um sistema que antecede em muitos séculos a eletrônica, e não pode incluir todas as novas notações requeridas pela música eletrônica.

Para muitos donos de micros, compreender a notação em pauta ou a MCL será a chave para obter o máximo da MIDI. Nas execuções ao vivo, essa interface é, antes de tudo, um meio de integrar vários sintetizadores, seqüenciadores, computadores rítmicos e baterias eletrônicas, tudo gerenciado por um computador na forma de um sistema. A maior preocupação dos músicos que fazem uso extensivo do seqüenciamento é a possibilidade de perderem a sincronização entre as diversas unidades do sistema, com um resultante "colapso musical". Executantes famosos, como os Thompson Twins e Howard Jones, são conhecidos por rodarem fitas com trilhas de fundo gravadas em estúdio enquanto fazem execuções "ao vivo", evitando assim o risco.

Pelo menos em teoria, grupos que usam multi-sintetizadores deveriam estar mais livres de problemas graças à MIDI. Uma característica desse avanço é que tais grupos não darão mais a impressão de serem do tipo "multi-sintetizador". No início da década de 70, os sintetizadores eram, em geral, instrumentos de teclado com tantos botões e potenciômetros deslizantes que mais pareciam um painel de avião. Hoje, se um sintetizador de teclado está usando a MIDI para controlar um segundo ou um terceiro sintetizador, não há necessidade de mais teclados além daquele já existente no instrumento mestre.

À medida que o uso da MIDI se generaliza, mais "módulos sintetizadores" aparecem no mercado. São simplesmente as unidades geradoras e seqüenciadoras de som que antes faziam parte dos instrumentos de teclado. Como esses módulos têm pouco ou nenhum interesse visual, não é necessário que estejam presentes no palco.

Há outro recurso anterior à MIDI, mas que deve ganhar mais atenção à medida que o número de teclados diminui. Trata-se da possibilidade de se sintetizar o som eletrônico por dados provenientes de instrumentos de cordas e também por dados de controle, da respiração e da boca, provindos dos instrumentos de sopro. Em comparação ao ato mecânico de simplesmente pressionar uma nota no teclado, tanger uma corda ou fazer vibrar uma palheta envolve a transmissão de informações acústicas mais complexas ao instrumento em questão. Se esses dados são codificados digitalmente e transmitidos via MIDI a um módulo sintetizador fora do palco, não há razão para que um saxofonista não controle a eletrônica de todo o seu grupo — até mesmo a da bateria. A Yamaha incorporou controle de respiração em seu sintetizador DX7, e a SynthAxe — uma guitarra digital desenvolvida há pouco tempo — foi projetada de modo a usar a MIDI para controlar a saída de um Fairlight.

Isso tudo significa que, acionando a tecla do DX7 e controlando os dados da respiração, podem-se produzir as características únicas de execução de um saxofone, e, de modo semelhante, articula-se um som de "trombone" Fairlight dedilhando-se uma guitarra. Embora nenhum desses recursos seja iminente — afinal, a SynthAxe é uma "guitarra" muito cara —, eles indicam as tendências prováveis para execuções ao vivo num futuro próximo. Talvez ocorra uma redução gradual do número de teclados no palco, ganhando importância os instrumentos de cordas, sopro e, possivelmente, de percussão afinados, tais como os vibrafones; e, à medida que baratear a tecnologia da amostragem sonora, o som acústico poderá tornar-se predominante.

Nesse rumo, o final da década de 80 trará de volta, para alívio dos tradicionalistas, conjuntos de estilo jazzístico compostos de guitarra, saxofone, baixo elétrico e bateria. Mas eles poderão ficar confusos ao observar o guitarrista tocando um vibrafone invisível ou o saxofonista emitindo sons de bateria com seu sopro.

Para muitos grupos acostumados a execuções ao vivo, a experiência inicial com um avançado estúdio de gravação pode ser intimidante. Eles vão tomar contato com sistemas operacionais que nunca encontraram antes, e um produtor que pode não conhecer muito bem o trabalho ou as intenções dos músicos. A gravadora, porém, espera que, nesse ambiente estranho, eles produzam versões "maiores e melhores" de seus sucessos no palco. Isso equivale a colocar um grupo de teatro semiprofissional num filme de grande orçamento e esperar um sucesso de bilheteria instantâneo. Às vezes a transição é bem-sucedida, e todos os envolvidos ficam satisfeitos. Mas, com freqüência, as idéias originais se perdem num labirinto de aparelhos de estúdio, e o grupo produz um dispendioso fracasso.

É bastante comum ocorrer ruptura quando o grupo abandona seu próprio equipamento — e, com efeito, "seu" som — para adotar os instrumentos do estúdio. Uma idéia que funcionou num sintetizador Mini-Moog pode não funcionar num computador de amostragem digital Fairlight. E, se esse tipo de fracasso musical ocorre com muita freqüência, a justificativa para o uso do estúdio começa a enfraquecer.

No entanto, se os músicos do grupo já estiverem familiarizados com a MIDI e empregarem um microcomputador para armazenar seqüências e outros dados de controle musical, os procedimentos nos estúdios avançados não serão desconhecidos. A nível mais imediato, seria possível experimentar idéias utilizando uma sucessão de diferentes instrumentos de estúdio com um mínimo de dificuldade, e com o conhecimento prévio de que as idéias e as seqüências podem ser transformadas pela simples troca dos sintetizadores.

Ter a MIDI como técnica de apoio mostra-se algo inestimável também quando se enfrentam outros sistemas de estúdio diferentes daqueles diretamente envolvidos com a geração de som. Uma mesa de mixagem que utiliza a lógica do estado sólido, por exemplo, possui um computador para uso exclusivo, que chamará e rodará novamente qualquer série de decisões tomadas nos estágios finais da gravação — o que é conhecido como mixagem. Quando toda a música tiver sido gravada em 24 trilhas separadas — guitarra em uma, vocais de fundo em outra, vocais principais em mais três, e assim por diante —, começa a tarefa crucial de balancear e mixar todos os elementos.

Em geral, é nesse ponto que cada elemento musical recebe tratamento com qualquer "efeito" desejado, para que sobressaiam ou se misturem. Pode-se acrescentar reverberação a uma nota de trompete, apenas num determinado ponto; com 23 outras coisas acontecendo ao mesmo tempo, é fácil esquecer esse detalhe. Utilizar, enquanto se está mixando, um computador para manipular tais incidentes se parece com um seqüenciamento de MIDI em grande escala.

Outra técnica — originalmente desenvolvida para edição e sincronização de vídeo — que vem sendo empregada na produção musical consiste no uso do código de tempo. Trata-se de um recurso semelhante a um relógio digital com um sinal de disparador, mas gravado em fita. Utiliza palavras de 80 bits para fornecer dados de sincronismo quando se grava uma trilha sonora sobre seqüências de vídeo, e torna possível seqüenciar conjuntamente eventos musicais e imagens na tela.

Como se vê, os músicos têm boas razões práticas para adquirir instrumentos compatíveis com a MIDI. Mais que isso, porém, é a MIDI constituir uma boa introdução aos avançados sistemas musicais atualmente em uso.

Arte eletrônica
Laurie Anderson, poetisa e artista de vanguarda de Nova Iorque, combina uma mistura inusitada de sons e equipamento sonoro com filmes, fitas magnéticas e vídeo, criando um estilo único. Em algumas de suas canções ela utiliza diretamente — ou como apoio de fundo — sons que vão desde sinos africanos aos instrumentos eletrônicos em "estado-de-arte", como o EMS Vocoder e o Synclavier, da New England Digital.

IDÉIAS EM ORDEM

Combinando recursos de processamento de texto e manipulação de listas, o programa BrainStorm — aqui analisado — proporciona flexibilidade na organização de idéias e de informações.

O BrainStorm, da firma inglesa Caxton Software, já foi cognominado de "processador de pensamentos", mas isso não significa que esse programa tente suplantar o cérebro humano. Na verdade, o BrainStorm ajuda a organizar os pensamentos. Para entender seu funcionamento, convém estabelecer uma analogia direta com os métodos pré-eletrônicos.

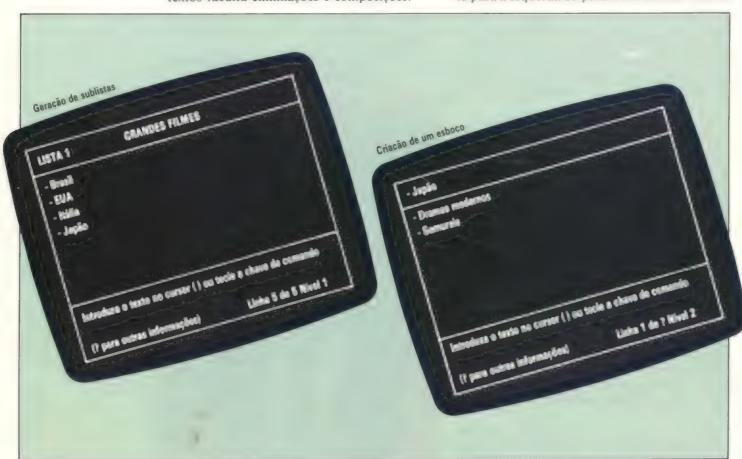
No desenvolvimento de qualquer projeto complexo, é comum a elaboração de listas de tarefas. Uma lista dos objetivos principais gera sublistas de como executar cada item arrolado, e assim por diante; o planejador pode ficar confuso com tantas folhas de papel. Algumas alterações na estrutura implicam tanto apaga-e-reescreve que o sistema se torna incontrolável. O BrainStorm simplifica tais alterações e acréscimos, da mesma forma que um processador de textos faculta eliminações e composições.

Qualquer item de uma lista ou sublista pode ser promovido a cabeçalho de uma sublista inferior; e, quando é necessário agrupar atividades similares de listas diferentes, elas poderão receber o mesmo nome e ser chamadas homônimas. Nesse caso, quaisquer introduções realizadas numa sublista encabeçada por um homônimo são agregadas a todas as outras listas que o usam.

Quem estiver familiarizado com os comandos de edição de um processador de textos, como o WordStar, achará muito fácil acostumar-se aos comandos de controle de cursor do BrainStorm. Digitando-se [CTRL] em conjunto com a letra [S], o cursor move-se para a esquerda. [CTRL]-[D], [CTRL]-[E] e [CTRL]-[X] fazem o cursor mover-se para a direita, para cima e para baixo, nessa ordem. Esses quatro comandos formam um padrão lógico no teclado, mas é possível redefini-los.

Mais difícil é acostumar-se com a possibilidade de elevar e baixar o cursor quando se introduz um texto, embora seja preciso alterar o modo — usando [CTRL]-[A] —, a fim de movê-lo para a esquerda ou para a direita numa linha.

Geração de sublistas

Criação de um esboço

Funcionamento do programa

O menu de abertura do BrainStorm oferece onze opções, escolhidas pela letra inicial do comando correspondente: Use, Load, Print, ID Drive (para utilizar outra unidade de disco), Save, Write (para gravar em disco), Directory, Xit (saída), Merge e Kill.

Inicialmente, o usuário digita [U] e passa a introduzir suas idéias como uma lista mais ou menos aleatória, chamada modelo. Por exemplo, um determinado modelo poderia consistir em:

```
Ler o manual
Digitar a lista
Digitar a sublista
Editar a lista
```

Os comandos de controle não aparecem imediatamente, mas são listados teclando-se [?]. Levando o cursor a qualquer item da lista e digitando [CTRL]-[R], promovemos esse item a cabeçalho de uma sublista. Da mesma forma, qualquer item dessa sublista pode transformar-se em cabeçalho de uma nova sublista, e assim por diante, até esgotar-se a memória. Digita-se [CTRL]-[C] para retorno à lista anterior.

Os itens tanto podem ser movimentados no interior de uma lista como transferidos para listas em outros níveis, quando rotulados com o caractere @. Para executar o movimento, digita-se em seguida [CTRL]-[G] (de Get) ou [CTRL]-[P] (de Put). Após a utilização de [CTRL]-[G], o caractere @ se move automaticamente para o item seguinte da lista; assim, um novo comando [CTRL]-[G] acessará o item seguinte, e assim por diante. Esse recurso permite transportar uma série de itens para sublista inferior ou superior.

Para inserir novos itens numa lista já existente, basta mover o cursor para o início da linha que deverá suceder o novo item e digitá-lo. Ao teclar [Return], o restante da lista descerá uma linha, dando espaço para o novo elemento. Com o comando [CTRL]-[A] altera-se qualquer item.

Itens homônimos de diferentes listas — por exemplo, datas — têm referência cruzada. Assim, se a lista exige a ocorrência de um evento em determinado dia — por exemplo, 1.° de janeiro —, podemos criar uma lista de eventos para tal data, que será automaticamente atualizada por meio da referência cruzada.

Pode-se criar qualquer número de homônimos e então acessá-los em seqüência, usando [CTRL]-[S] para a ocorrência seguinte e [CTRL]-[D] para a anterior. Após a última ocorrência, [CTRL]-[S] volta em loop para a primeira.

Impressão de listas

É possível imprimir as listas, normalmente formatadas com recuos para indicar seus níveis. Por exemplo:

```
Ler o manual
  Tirar o manual da caixa
    Consultar o sumário
      Encontrar a página desejada
      Ler a página
    Recolocar o manual na caixa
Começar a digitar a lista
  Digitar [CTRL]-[R] para promover o item da
    lista a novo cabeçalho
    Digitar sublista
  Digitar [CTRL]-[C] para voltar à lista anterior
Editar a lista
  Digitar [CTRL]-[A] para alterar uma entrada
```

Também é possível editar as listas fora do BrainStorm. O WordStar e outros programas de processamento de texto as reconhecerão como documento se estiverem gravadas em disquetes. Nesse caso, podem-se editá-las, imprimi-las e, se necessário, regravá-las. Significa, por exemplo, a possibilidade de empregar o BrainStorm para preparar a sinopse de um livro, que seria escrito então com um processador de texto comum, acessando-se o arquivo BrainStorm.Doc enquanto o livro avança.

Devido à flexibilidade do programa, esse processo pode começar com as primeiras anotações de idéias básicas — mais uma vez, algo normalmente feito em pedaços de papel —, que então se tornam títulos de capítulos, sob os quais se escreve a história. Se um item de um capítulo crescer de forma a merecer transformar-se em capítulo à parte, isso será feito com facilidade. Da mesma forma, títulos de capítulos que acabam se revelando menos importantes do que se imaginava podem ser "rebaixados" a subitens.

Assim, o esboço rudimentar do livro começa a configurar-se e, uma vez que o WordStar e pacotes semelhantes podem acessar os arquivos do BrainStorm, a transformação desse esboço inicial no manuscrito acabado flui naturalmente, a partir dos primeiros processos de reflexão. Isso também quer dizer que não é necessário adotar os procedimentos trabalhosos exigidos pelo BrainStorm se se deseja, por exemplo, imprimir com espaço duplo ou triplo. Douglas Adams, autor de *The hitchiker's guide to the galaxy (Guia do caminhante na galáxia)*, usou o BrainStorm dessa maneira para desenvolver o jogo de aventura baseado em seus livros.

O programa vem com um exemplo, que inclui a estrutura de uma agenda, um arquivo de nomes e endereços, uma lista de tarefas (com as sublistas "urgente", "importante" e "não esquecer"), além de uma seção para anotações. É de muito valor, podendo ser anexado aos modelos do usuário usando-se a opção Merge.

Aprende-se facilmente a usar o BrainStorm. O manual é muito claro (foi escrito com o próprio BrainStorm), dispensando recorrer-se a ele constantemente, uma vez que os comandos estão sempre à disposição num menu de tela.

Nem todos os comandos são mnemônicos, de forma que alguns mostram-se difíceis de memorizar. Porém o BrainStorm inclui o programa Installb, muito simples de usar, que permite reconfigurar todos os comandos e alterar todos os menus para adequação ao novo esquema determinado pelo usuário.

Listagem de porte

O BrainStorm combina um processador de texto com recursos de manipulação de listas encontrados nas linguagens LOGO e LISP. Qualquer item de uma lista pode tornar-se o cabeçalho de uma sublista, o que facilita o esboço e a organização de tarefas hierárquicas. No caso de cabeçalhos homônimos de sublistas, o BrainStorm manterá apenas uma lista desse nome e a acessará sempre que o homônimo for chamado, independentemente da lista em que for encontrado.

```
LISTA 1
GRANDES FILMES
1) Brasil
2) EUA
3) Itália
4) Japão

LISTA 1.1
BRASIL
1) Vencedores de festivais
2) Documentários
3) Co-produções

LISTA 1.2
EUA
1) Dramas
2) Comédias
3) Guerra

LISTA 1.3
ITALIA
1) Neo-realismo
2) Comédias

LISTA 1.4
JAPÃO
1) Dramas modernos
2) Samurais

HOMÔNIMO 1
VENCEDORES DE FESTIVAIS
1980 - Gaijin
1980 - Tudo bem
1981 - Pixote
1982 - Asa branca
1982 - Das tripas coração
1983 - Pra frente Brasil
1983 - Raoni
```

ROBÔS NA TRILHA

Robôs e labirintos

Robôs...
As competições de Micro Mouse (em que mouses-robôs fazem o possível para vencer um labirinto) são uma valiosa fonte de conhecimento prático e de habilidade técnica para os aficionados da robótica.
O Quester, de David Buckley, possui uma gama completa de sensores: ópticos, sônicos e sensíveis ao toque.

... e labirintos
Cada Micro Mouse possui um período de prática para "aprender" o esquema do labirinto, utilizando qualquer método que não exija comunicação externa.
Em seguida, deve atravessar o labirinto, numa luta contra o relógio — o objetivo básico é atingir o centro no menor tempo possível.

Existem três métodos principais para conseguirmos movimentar um robô, e os motores mais usados são os elétricos. Mas, se ele deve fazer aquilo que queremos, como controlar seus movimentos?

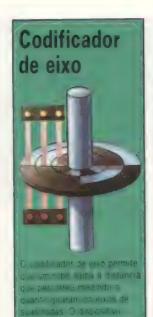
O método mais simples de movimentarmos um robô consiste num dispositivo mecânico que "lê" um cartão nele inserido, criado especialmente para essa tarefa. O perfil do cartão é seguido por uma came que comanda as alavancas responsáveis pelo direcionamento do robô. No passado, chegaram a ser construídas miniaturas de carros e robôs de brinquedo que funcionavam segundo esse princípio. Criava-se um programa utilizando uma tesoura para cortar o cartão na forma desejada, e o robô se movimentava conforme a margem recortada.

Outros robôs eram dotados de dispositivos que lhes permitiam seguir uma direção graças a relés eletromecânicos internos. Esses métodos, porém, tinham aplicações limitadas, pois as partes mecânicas costumam ser caras e um tanto imprecisas. Apesar disso, forneceram as bases para os métodos contemporâneos.

Um dos melhores meios hoje empregados consiste numa trilha traçada no solo para o robô seguir. Trata-se de um sistema semelhante ao usado para miniaturas de carros de corrida, que possuem um pino-guia inserido numa fenda contínua na pista. São dois os tipos mais comuns de robôs que seguem trilhas: os orientados por uma linha traçada no chão e os guiados por um fio.

Para seguir uma linha, o robô utiliza um elemento fotossensível (célula fotoelétrica ou sensor infravermelho), que determina se ele está numa área "iluminada" ou numa área "escura". Por exemplo, se a cor do chão for escura e a linha clara, a saída do sensor terá intensidade máxima toda vez que ele estiver sobre a linha. Assim, seguindo o caminho que proporciona essa máxima intensidade, o robô nunca se desviará da linha.

Mas essa técnica apresenta um problema: o que deve fazer o robô quando a saída do sensor

Codificador de eixo

O codificador de eixo permite que um robô saiba a distância que percorreu, medindo o quanto giraram os eixos de suas rodas. O dispositivo consiste num disco calibrado adaptado a um eixo. A placa circular tem vários anéis concêntricos, cada um traçado em áreas transparentes ou opacas. A posição do eixo é determinada por uma fonte de luz e um fotossensor.

A precisão do codificador de eixo depende do número de anéis. A ilustração mostra um com três anéis, capaz de codificar os números binários de 000 a 111 (decimais de 0 a 7). Esse codificador oferece uma precisão de 360/8 = 45 graus. Oito anéis ofereceriam 1,41 grau.

sofre uma queda, indicando que ele abandonou a linha? Com um sistema único, o máximo que pode fazer é vagar de um lado para outro até que a saída do sensor retorne aos níveis anteriores, indicando que ele voltou à linha. Esse método não se verifica tão aleatório quanto parece. Por exemplo: se o robô se dirigia para a esquerda no momento em que a queda foi acusada, é razoável que ele vá para a direita a fim de reencontrar a linha. E, se a reencontrar, é razoável também que ele entenda estar a nova direção a seguir entre a rota que seguia antes (esquerda) e a que teve de seguir depois (direita).

Um sistema para reduzir o tempo que um robô "descarrilado" leva para reencontrar a direção certa faz uso de dois sensores, cada qual apontado para um dos lados da linha. Isto é: quando o robô anda na linha, a saída de ambos os sensores será baixa. Se começar a se desviar, a saída de um dos sensores acusará. Em outras palavras: ele saberá imediatamente que andou errado e em que direção. Se se desviar para a direita, o elemento esquerdo apontará, e o robô interpretará isso como um sinal para retornar à esquerda, o que o colocará de novo no caminho certo.

Para esse sistema, não é indispensável uma linha branca contra um fundo escuro, pois o contrário funciona da mesma forma. O importante é o contraste, e também que o programa instrua o robô quanto ao que fazer quando um sensor ler valores incorretos.

Outro sistema empregado em robôs que seguem trilhas consiste numa corrente elétrica percorrendo um fio estendido no chão. Essa corrente gera ao redor do fio um campo magnético, o qual é detectado pelo sensor. E este não precisa ser, necessariamente, complicado: uma pequena bobina de fio é suficiente para captar o campo magnético. Sob a ação do campo, resulta na bobina uma tensão, que, amplificada, atua como os fotossensores. Robôs industriais que se movimentam em várias direções são em geral orientados por um fio embutido no chão. Se seguissem uma linha pintada, tudo correria bem enquanto o chão não ficasse sujo.

Controle remoto

Outro método envolve um operador humano controlando o robô a distância. É particularmente útil quando as tarefas confiadas ao robô são desempenhadas em ambiente hostil ou perigoso ao ser humano. Exemplos: desativação de bombas; manipulação de produtos químicos perigosos ou materiais radioativos; trabalho em locais muito quentes ou muito frios, ou que, simplesmente, ofereçam grandes riscos. Um famoso robô desse tipo foi o soviético Lunokhod 1, levado à Lua pela nave espacial Luna 16, em 1970. Era um robô sobre rodas que colheu informações sobre o solo lunar, controlado por ondas de rádio enviadas da Terra.

O comando de robôs como esse não difere muito do de um aeromodelo radiocontrolado. Há dois tipos de sinais de rádio: o analógico, que varia de intensidade conforme a quantidade de movimentos exigidos do robô; e o digital, configurador de um padrão de bits que dá detalhes dos movimentos a serem executados. As comunicações analógicas em geral têm menos êxito que as digitais. Isso porque vários fatores podem interferir na intensidade dos sinais numa trans-

Fotossensores

Com o auxílio de um elemento fotossensor, é possível projetar robôs para seguirem trilhas no chão. Estas podem ter cor clara sobre fundo escuro ou cor escura sobre fundo claro.
Nos dois casos, utilizamos uma célula fotoelétrica para detectar mudanças no brilho da superfície.

Com um único sensor, o robô saberá apenas se está sobre a trilha ou fora dela. Se sair da demarcação, terá de procurá-la sem rumo. Com um sistema de duplos sensores, o robô sabe que está no caminho certo quando ambos detectam o fundo claro em cada lado da trilha. Se ele se desvia, um dos sensores acusa um aumento na intensidade luminosa, detectando o afastamento.
O robô então sabe como voltar à trilha de acordo com o sensor que foi ativado.

missão analógica. Ouça uma emissora de rádio distante e perceba como a recepção varia conforme a hora do dia e as condições atmosféricas. Pois o mesmo tipo de problema pode afetar as comunicações com os robôs.

Os métodos digitais também apresentam dificuldades, principalmente quando uma interferência provoca perda de bits ou insere bits em posições indevidas. Para evitar isso e operar com maior segurança, repetem-se as mensagens ao robô, e ele só começa a agir depois de receber várias vezes a mesma mensagem.

Sistemas de realimentação

Uma técnica mais sofisticada consiste em utilizar um sistema de loop (circuito completo), no qual o robô dá um retorno ao transmissor das informações que acabou de receber. Seria uma espécie de diálogo entre transmissor e robô. Por exemplo, o transmissor manda: "Mova-se para a frente"; ao receber essa mensagem, o robô pergunta: "Você disse 'Mova-se para a frente'?"; o transmissor responde: "Sim"; só então o robô se move. Isso evita erros desastrosos quando, digamos, o robô manipula lixo atômico ou está prestes a despencar para dentro de uma cratera na Lua.

Essas mesmas técnicas gerais aplicam-se a outros tipos de controle remoto. Por exemplo: certos robôs podem ser controlados a distância por emissores infravermelhos do tipo utilizado no controle remoto de aparelhos de tevê; por ultrasons ou por um som de natureza inconfundível, como o de assobios ou palmas. Seja qual for o método empregado, as técnicas básicas de transmitir a mensagem e confirmar que o robô a recebeu são as mesmas.

Se o robô trabalha bem próximo do operador, tornam-se desnecessárias técnicas tão sofisticadas: podem-se transmitir as ordens através de um fio. Existe ainda a possibilidade de se utilizar mais de um fio, o equivalente a uma série de canais num aeromodelo por radiocontrole. No caso do robô, porém, em geral usam-se os fios para proporcionar comunicações, não seriais, mas paralelas (envia-se um string de bits em paralelo, através de todos os fios, e não uma série de pulsos através de um único fio). Desse modo, a comunicação se faz mais rápida. Mais importante talvez seja o fato de a maioria dos computadores possuir uma saída paralela, que constitui um meio conveniente de transmitir instruções ao robô.

Se o operador controla os movimentos do robô pelo teclado de um computador e se pode vê-lo, então, em princípio, não há grande diferença se o controle for exercido pelo operador ou pelo computador. Isso porque, tal como no caso do aeromodelo controlado por rádio, o operador sempre vê o que o robô faz e pode corrigir seus erros prontamente. Se, porém, o robô estiver a alguma distância (na Lua ou mesmo na sala vizinha) ou se for controlado por um programa de computador e não por comandos introduzidos pelo teclado em tempo real, então ele terá de ser um pouquinho mais inteligente. Precisará, essencialmente, de alguma forma de realimentação (feedback).

Esse processo capacita-o a ajustar o que está fazendo àquilo que já fez e com o que deveria fazer. Digamos que você queira mover o robô 1 m; se o estiver controlando diretamente, você poderá iniciar o movimento, avaliar o avanço e pará-lo assim que o metro tiver sido percorrido. Isso ocorre porque você tem uma realimentação visual sobre o robô: vê até onde ele foi, até onde quer que ele vá e assim pode corrigir-lhe as ações.

Na ausência de realimentação sensorial humana, o robô deve ter um pouco da sua própria, para poder movimentar-se com precisão. O tipo que segue uma linha utiliza a realimentação da linha; do mesmo modo, o que é controlado por computador precisa usar alguma forma de realimentação para que seu avanço seja exatamente de 1 m. O método mais comum de realimentação consiste no codificador de eixo, que associa as rotações dessa peça ao deslocamento linear da máquina. Trata-se de um disco preso aos eixos principais das rodas do robô e que mede quanto elas rodaram. Assim, quando o computador o instrui para avançar 1 m, ele, ao mesmo tempo que se move, monitora os sinais enviados pelos codificadores para saber quanto andou. Depois de avançar até onde precisa, o robô pára. Caso ultrapasse o limite, retrocederá à medida certa, calculada com base nas informações enviadas pelos codificadores.

Um passo gigantesco
Em 1970, o Lunokhod 1 pousou na Lua com a missão de colher informações sobre a superfície do solo. Embora não fosse um autêntico robô, ele pôde transportar uma enorme carga científica.
Como qualquer objeto espacial sob controle remoto, o Lunokhod tinha a desvantagem de um atraso de 3 segundos entre sua transmissão de informações à Terra e o retorno de um comando em resposta.

WREN EXECUTIVE

Portátil, com duas unidades de disquete, modem de chamada automática incorporado e o BASIC da BBC, o Wren Executive volta, depois de alguns tropeços, a procurar seu espaço no mercado.

Pássaro incomum
O Wren Executive é uma máquina portátil barata, com a característica incomum de rodar o BASIC da BBC, assim como o sistema operacional CP/M. Entre outros recursos, há duas unidades de discos, monitor monocromático e modem embutido.

O computador Wren Executive tem uma história acidentada. Construído pela firma inglesa Thorn EMI, era comercializado originalmente pela Prism. Quando esta abriu falência, no início de 1985, parecia que o Wren seria mais uma vítima da turbulenta indústria de informática. Mais tarde, porém, a Opus Supplies encarregou-se da distribuição do computador, planejando relançá-lo. Este artigo analisa as possibiljdades da máquina no competitivo mercado profissional.

O Wren Executive foi concebido como máquina portátil, ou seja, um equipamento auto-suficiente constituído por uma CPU, um monitor de vídeo e duas unidades de discos. Na parte traseira da máquina há uma alça de transporte. Como muitos outros portáteis, o Wren mostra-se um tanto pesado para transporte por longas distâncias e talvez seja mais adequado como computador de mesa, que pode ser eventualmente mudado de uma área de trabalho para outra.

O teclado é de configuração QWERTY e apresenta como única diferença os símbolos * e #, que têm suas próprias teclas. À esquerda, encontram-se cinco teclas de função programáveis — operadas em conjunto com a [Control] e a [Shift], elas proporcionam até quinze funções diferentes. Essas teclas possuem diversas finalidades, dependendo da aplicação, embora seu uso predominante se restrinja aos comandos CP/M simples como PIP e RENAME. À direita estão as quatro teclas de movimentação do cursor e a [Home], para retorná-lo ao canto superior da tela.

Medindo 150 x 105 mm, com resolução de texto de 80 x 25 caracteres, sobre fundo âmbar (em lugar do verde usual), a tela favorece longos períodos de trabalho. À direita ficam as duas unidades de discos flexíveis de 5 1/4 polegadas.

Um dos problemas do vídeo é o desenho do gabinete. Além de sua pequena altura, o equipamento não possui pés reguláveis para inclinação, dificultando a visualização da tela. Esta fica embutida no gabinete, e tem sua parte superior um tanto obscurecida por uma aba de 40 mm que se projeta sobre ela. Embora não se trate de um problema grave, traz algum incômodo ao usuário.

O Wren é bem provido de interfaces. Há várias saídas na parte traseira, permitindo sua conexão com uma ampla variedade de periféricos. Diferentemente de muitas máquinas comerciais na mesma faixa de preço, o Wren incorpora um modem de chamada automática. Seu proprietário, portanto, pode acessar todas as principais bases de dados sem precisar adquirir qualquer outro equipamento. Acompanha o Wren um cabo para ligação ao sistema telefônico. O software de comunicações fornecido com o equipamento permite armazenar vários números de telefone e chamá-los automaticamente quando necessário.

Ao lado da saída de comunicações existem duas outras, destinadas aos conectores de controles externos, por exemplo, de mouses e joysticks. Ao lado encontra-se um conector para saídas seriais de comunicações RS232C, que costumam ser usadas em computadores profissionais para modems externos. Como tal utilização não é uma necessidade no Wren, essas saídas são mais empregadas para redes locais e impressoras seriais.

Uma interface paralela padrão Winchester permite conexão a uma unidade de discos rígidos, que acrescenta de 0,5 a 50 Mbytes à capacidade de armazenamento. Há também uma interface paralela para impressoras compatível com o padrão Centronics. A máquina dispõe também de saída para monitor colorido RGB. Entre as interfaces para impressora e para monitor, há um diminuto botão [Reset], para a reinicialização do sistema.

Riqueza de interfaces
Uma das melhores características do Wren é a quantidade de interfaces existentes na configuração padrão. Além das interfaces comuns, Centronics e RS232C, há também uma para linha telefônica com modem incorporado. Existe, ainda, uma saída paralela para unidade de discos rígidos tipo Winchester.

Fundo útil
O monitor e as unidades de discos são protegidos, durante o transporte, por uma tampa de plástico reforçado, que se encaixa à parte frontal da máquina. Seguindo a filosofia "ligar na tomada e sair rodando", o fabricante tomou a atitude incomum de fornecer um cabo de alimentação com a tomada já encaixada. Esse componente fica acondicionado num compartimento da tampa durante o transporte.

Promessa de perfeição
O Wren é fornecido com seu próprio conjunto de softwares — Perfect Writer, Perfect Filer e Perfect Calc. Também há um programa genérico, o Executive Desktop, um software de comunicações, e o BASIC da BBC.

Monitor
O Wren dispõe de vídeo monocromático (âmbar), embutido, brilhante e fácil de ler.

Saída serial RS232C
Faculta a comunicação direta com outros computadores e outras aplicações seriais.

Saída para monitor colorido
Se o usuário desejar fazer uso dos recursos de cor do Wren, há uma saída para monitor RGB externo.

Alto-falante
Os recursos de som provêm do alto-falante do próprio Wren.

Interface para discos rígidos
Conecta uma unidade de discos rígidos tipo Winchester, ampliando bastante a capacidade de armazenamento do computador.

Chips de RAM
O processador tem apenas 8 bits, mas o Wren endereça até 400 Kbytes de RAM.

Unidades de discos
O computador vem equipado com duas unidades de discos flexíveis de 5 1/4".

Alça para transporte
Apesar da aparência desajeitada, o Wren é mais fácil e cômodo de transportar que muitos outros computadores na categoria dos portáteis.

Saída para impressora
Saída para interface paralela padrão Centronics, conecta uma impressora ao computador.

Saídas para paddles
Para controle por dispositivos externos — paddles, canetas ópticas.

Modem
Diferentemente de muitos outros computadores profissionais, o Wren já dispõe de um modulador de comunicações embutido.

O Wren baseia-se no conhecido microprocessador de 8 bits Z80B. Talvez seja esse o ponto mais fraco do computador. No momento em que os fabricantes de micros domésticos competem furiosamente no desenvolvimento de máquinas de 16 bits, o Wren parece um tanto antiquado em seu processamento. Embora de rapidez razoável para os padrões de 8 bits, seu desempenho não se compara ao dos micros de 16 bits, como o IBM PC e o Macintosh da Apple.

Como o Wren faz uso do processador Z80, era natural que seus idealizadores decidissem pelo CP/M como sistema operacional. Pelos padrões recentes, no entanto, o CP/M deixa muito a desejar quanto à interação com o usuário, apesar de sua potência. Os projetistas minoraram esse problema incorporando ao sistema operacional um procedimento comandado por menus, o que permite a escolha de um aplicativo por meio do cursor e da tecla [Return]. O sistema, então, espera a inserção do disco apropriado e executa o aplicativo de modo automático. Assim, muitos perceberão que não há necessidade de usar diretamente o CP/M, já que quase tudo o que o usuário comercial médio precisa está incluído no

Os programas fornecidos com o computador são o sistema CP/M, o processador de texto Perfect Writer, o Executive Desktop (um pacote para controle de agendas — datas, compromissos etc.), o banco de dados Perfect Filer, a planilha financeira Perfect Calc e um disco contendo software de comunicações e o BASIC. Embora a máquina tenha uma implementação quase total da linguagem, por se basear no processador Z80 e não no 6502, surgem algumas diferenças, principalmente nas rotinas de entrada e saída. A implementação da linguagem também parece ter gerado alguns erros. Por exemplo, nos modos 0 e 1, o cursor desaparece da tela. A tecla [Delete] não funciona nesses modos, e o retrocesso do cursor bem como a tentativa de reescrita resultam na superposição do novo caractere ao anterior. Isso impede uma programação eficiente nesses modos, embora os outros pareçam funcionar bem.

Em última análise, o sucesso do Wren Executive depende de ele encontrar seu espaço correto no mercado. Com modem embutido, sistema operacional CP/M, BASIC da BBC e um custo baixo, o Wren parecia valioso quando de seu lançamento, no primeiro semestre de 1984. Mas o potencial de vendas das máquinas profissionais de 8 bits, mesmo na faixa inferior do mercado, diminuiu muito de lá para cá. Além disso, a máquina ainda é muito cara para atacar seriamente o mercado dos micros domésticos.

Sobra então o mercado educacional, onde seu emprego pode ser maior. O uso do BASIC da BBC, de ampla utilização nas escolas britânicas, combinado ao modem embutido — capaz de acessar as várias bases de dados que vão se tornando disponíveis —, poderia interessar os estabelecimentos educacionais que tentam obter o máximo retorno para seus investimentos.

WREN EXECUTIVE

MICROPROCESSADOR
Z80B

CLOCK
6 MHz

SISTEMA OPERACIONAL
CP/M

VIDEO
Modo texto: 80 × 25 caracteres; modo gráfico: 640 × 256 pixels

TECLADO
No padrão QWERTY, com 57 teclas, sendo cinco de funções programáveis e cinco de controle do cursor

LINGUAGENS DISPONÍVEIS
Basic da BBC

INTERFACES
Saída para monitor, interface padrão Centronics, saída serial RS232C, linha telefônica, duas saídas para paddles, interface para discos rígidos

DOCUMENTAÇÃO
Bem apresentada, oferece ao usuário explicações completas. Além do manual do usuário, mais um para cada aplicativo fornecido.

AVENTURAS E SIMULAÇÕES

Bastante divertidos, os jogos baseados em estratégias que simulam situações da vida real testam nossa capacidade de planejamento, bem como outras, que empregamos no dia-a-dia.

Além de proporcionar jogos que exigem reflexos e movimentos rápidos para os fãs do "mata-marcianos", o computador é um excelente meio para um desafio diferente. Trata-se do jogo de simulação, que requer habilidades muito diferentes das exigidas pelos jogos tipo fliperama. A "simulação" consiste em criar na memória do microcomputador modelos da vida real para os propósitos do jogo. Esses modelos podem ser simuladores em "tempo real", como os programas de simulação de vôo, ou baseados em estratégias.

Nestes, o jogador, ou grupo de jogadores, recebe uma série de tarefas a realizar e algumas informações que o ajudam na tomada de decisões. Um exemplo típico é o clássico Banco Imobiliário, no qual os participantes recebem dinheiro com o objetivo de aumentar seus lucros. O que distingue um jogo de simulação dos de aventura e ação é que, neste último, o jogador encontra armadilhas e outros riscos sem saber exatamente quando surgem; já numa simulação, em geral ele pode prever o que está por vir, direcionando seus recursos para vencer contingências esperadas.

Esse tipo de jogo, portanto, exige de seus participantes capacidade de organização e previsão; em outras palavras, habilidade de planejar com antecedência, em vez de decidir rápido sobre uma situação imediata. Para refletir o mundo real, nem tudo num jogo de simulação deve ser deixado correr exatamente como planejado. Uma boa simulação sempre introduz fatores aleatórios (na verdade, pseudo-aleatórios) para frustrar os esquemas do participante. Um bom jogador deve, portanto, considerar o fator "acaso" a fim de minimizar os danos causados por eventos não previstos.

Os jogos de simulação têm a vantagem de permitir que as pessoas joguem em equipe, combinando inteligência e poder de decisão contra o computador, para cumprir uma tarefa predeterminada. Por essa razão, esses jogos começam a encontrar larga aceitação em diversas escolas, pois possibilitam o aperfeiçoamento do tipo de comportamento que se espera dos alunos no mundo adulto do trabalho, combinando recursos e talentos de um pequeno grupo para atingir um certo objetivo.

Além de todos esses atributos, os jogos de simulação projetam-se numa área raramente tocada pela maioria das aplicações computacionais: a das questões éticas e morais. Por exemplo, os jogadores podem ser levados a ponderar a respeito do lucro crescente contra o bem-estar dos empregados. Quando se tomam tais decisões em equipe, costumam ser precedidas de debates acalorados sobre o certo e o errado.

Muitas situações diferentes prestam-se bem à simulação em computador. Geralmente o fator comum está em envolverem a realização de uma determinada tarefa, a fim de se atingir um objetivo específico. Em certos casos, este pode ser escolhido pelo jogador. A meta consiste em permanecer nos negócios por um certo período de tempo ou, alternativamente, ganhar um bilhão de cruzeiros. Muitas simulações se baseiam em estratégias financeiras, mas outras têm objetivos mais filantrópicos.

Um jogo de simulação atualmente utilizado nas escolas primárias inglesas constitui um projeto para localizar e recuperar os destroços do galeão elizabetano *Mary Rose*. Os jogadores recebem também documentos que complementam o programa e dão pistas adicionais que ajudam no processo de tomada de decisões. Num outro jogo, de simulação mercantil, os participantes têm de equipar e despachar um navio para o No-

Mundos imaginários
Abaixo estão três exemplos dos principais tipos de jogos de simulação. O Space Shuttle é uma simulação realista de uma viagem do ônibus espacial. No Air Traffic Control, uma simulação em tempo real, você dá instruções a vários aviões para que possam decolar e aterrissar com segurança. E The Great Nordic War simula a Guerra dos Trinta Anos, com o jogador desempenhando o papel de Carlos XII da Suécia.

Simulação de serviços

As simulações têm grande utilidade na projeção do fluxo de tráfego e de sistemas de serviços, onde a natureza do modelo as torna ideais para uso em computadores. Tais simulações normalmente se baseiam em princípios estatísticos e matemáticos que prevêem distribuições de freqüências das variáveis usadas.

Os princípios de uma simulação podem ser demonstrados observando-se um sistema de serviços simples como o metrô. O esquema existente permite acesso à plataforma de embarque de duas maneiras: pagando diretamente na bilheteria, ou pelos bloqueios automáticos, que são operados por bilhetes comprados antecipadamente. O fluxograma ilustra os processos envolvidos na produção de uma simulação simples do sistema. São usadas três distribuições: uma para a chegada dos passageiros, outra para o tempo de atendimento na bilheteria e uma terceira para o método de entrada (manual ou automático).

As duas primeiras distribuições derivam de observações estatísticas e métodos matemáticos; a terceira variará a cada passagem do programa, para se observarem as ramificações das várias proporções de bloqueios manuais e automáticos. Essa simulação simples calculará o tempo médio de espera dos passageiros que entram no sistema.

SISTEMA MANUAL, COM FITA

BILHETERIA

EMBARQUE

$ 30 | $ 75 | $ 95

SISTEMA DE ENTRADA AUTOMATICA

Início

Inicializar

Tempo terminado ?

S

Calcular e emitir os resultados

Parar

N

Tempo alocado = o menor entre o tempo de chegada e o tempo de atendimento na bilheteria

Diminuir o tempo de chegada do tempo alocado

Diminuir o tempo de atendimento na bilheteria do tempo alocado

Incrementar o tempo de espera total pelo tempo alocado x comprimento da fila

O passageiro chegou?

S

Distribuição amostra da entrada

A entrada é automática?

N

Incrementar o número da fila

S

Incrementar o número dos que entraram no sistema

O passageiro na bilheteria foi atendido?

S

Incrementar o número dos que entraram no sistema

N

A bilheteria está livre?

S

Há mais gente na fila?

S

Decrementar do número de pessoas na fila

Distribuição amostral de atendimento na bilheteria

N

N

Esta simulação não é em tempo real, mas calcula o tempo para cada evento toda vez que o loop é executado. Usa-se essa "fração" de tempo — ou "tempo alocado" — para atualizar as outras variáveis. O emprego do tempo alocado permite que o sistema simulado funcione com muito mais rapidez que um sistema real: uma hora de simulação pode equivaler a um dia de operação do sistema real.

vo Mundo, no século XV. Estruturado como uma série de módulos interligados, seu programa é elaborado de forma tal que o cenário seja facilmente alterado, proporcionando uma base para diferentes contextos. Assim, a simulação pode ser situada em outras épocas, transformando-se, por exemplo, numa jornada de ficção científica para incremento do comércio com um planeta distante.

O objetivo do jogo, para um ou mais jogadores, é comercializar bens para maximizar o lucro, contrabalançando esse propósito com a segurança e o conforto da tripulação. O jogo se divide em três partes: preparação, viagem e comércio. A primeira fase envolve a obtenção de recursos e o planejamento da viagem; a segunda introduz elementos aleatórios no jogo. A habilidade em lidar com essas contingências dependerá muito de como se executou o planejamento durante a primeira fase do jogo. A terceira fase envolve o estabelecimento de comércio lucrativo e de boas relações com os habitantes locais.

As principais tarefas do programa consistem em acompanhar as decisões dos jogadores, criar diversas situações imprevisíveis, analisar e monitorar a condição dos participantes conforme o desenrolar do jogo.

REDES LOCAIS

Periféricos e softwares de qualidade, mas dispendiosos, podem ser compartilhados por diversos usuários, permitindo reduzir os custos e estabelecer uma rede de intercomunicação.

As impressoras e as unidades de discos de qualidade exigem elevado investimento de capital, só justificável quando sua utilização é intensa. Assim, ao cogitar a compra de um periférico, o empresário muitas vezes acaba desistindo devido ao tempo ocioso do aparelho. No entanto, na execução de determinadas tarefas, um equipamento desses pode ser de inestimável valor.

Uma unidade de discos tipo Winchester, por exemplo, revela-se de enorme importância em trabalhos que necessitam de acesso rápido à informação. Mas sua capacidade de memória pode ser excessiva para a maioria dos proprietários de micros. Esse problema se resolve pela implantação de redes de computadores, que permitem a um grupo de usuários compartilhar recursos de hardware, software e dados.

Esse uso conjunto de dados, quando efetuado através da interligação de computadores, estações, serviços, periféricos e congêneres numa área local, isto é, num mesmo escritório, prédio ou conjunto de prédios, origina a chamada rede local. Na medida em que organizações como bancos, indústrias, hospitais e escolas começam a aumentar seu número de computadores, torna-se necessária a interligação para compartilhar recursos e informações.

Tipos de rede

Estrela
Nesse tipo de rede, o controle central, ao qual se liga o operador de cada máquina, comanda os periféricos compartilhados.

Barramento
Nessa disposição, os dados e as mensagens de controle passam diretamente de um usuário para outro.

Anel
Menos usada que as outras, essa configuração conecta os elementos em sequência. A informação chega ao destino após percorrer mais de metade da rede.

Degradação lógica

Para se ter uma idéia do que seja a degradação lógica num sistema de rede, consideremos diversos usuários compartilhando o mesmo banco de dados, representado por uma unidade de disquetes e uma impressora, ligados diretamente a um dos micros. Se, por exemplo, todos os micros necessitam buscar uma informação na unidade de disquete simultaneamente, a CPU do elemento que tem os periféricos precisa compartilhar seu tempo para atender às pesquisas. Isso resulta num tempo de espera em cada micro — é a degradação lógica.

O problema se mostra bem maior no caso de unidades de disquetes do que no de impressoras. No entanto, o sistema de acesso às informações pode ser aprimorado utilizando-se discos rígidos, que permitem reduzir o tempo de resposta.

A utilização das redes locais varia conforme a definição das necessidades. Por exemplo, em determinada empresa, a rede pode controlar o fluxo de caixa, estoque, folha de pagamento, emissão de notas fiscais e a contabilidade geral. Já em outra, pode servir para compartilhar dados oriundos de computadores de grande porte, processando-os independentemente.

Além da questão do custo dos periféricos, a interligação de equipamentos em rede estimula o desenvolvimento de todo o sistema sobre uma mesma linha de aplicativos, o que resulta em economia de software. As redes locais permitem ainda o aumento dos recursos físicos (periféricos) disponíveis para cada estação (usuário), além de maior interligação entre aplicações e confiabilidade elevada a baixo custo. Soma-se a isso a possibilidade de crescimento gradativo conforme as necessidades da organização.

Uma rede local executa algumas tarefas básicas, como, por exemplo:

a) Transferência de arquivos.

b) Teletipo — transmissão de mensagens de tamanho fixo (512 bytes).

c) Comunicação entre processamentos distintos.

d) Conversão de códigos de comunicação.

e) Detecção e recuperação de erros na transmissão.

f) Conversão de velocidade de transferência.

Para a realização dessas tarefas, são necessárias algumas rotinas escritas em linguagem de baixo nível (ASSEMBLY), que disciplinam o acesso das comunicações ao meio de transmissão.

O conjunto dessas rotinas chama-se protocolo. Os dois tipos de protocolo mais utilizados são o TOKEN, "passagem por permissão", e o CSMA (Carrier Sense Multiple Acess) "por contenção". No TOKEN, o dado transmitido é analisado pelas máquinas, numa de cada vez, e retransmitido às demais que compõem a rede quando seu endereço corresponde ao de outra máquina. No CSMA, ao contrário, todas as máquinas podem acessar o meio ao mesmo tempo e só receber a informação a elas destinada.

O que determina, na grande maioria das vezes, o tipo de protocolo é a disposição física da rede, ou seja, o tipo de interligação existente entre os equipamentos, também conhecida como arquitetura da rede. Os três tipos mais difundidos são a rede em anel (ring, ou loop), em barramento global (bus) e em estrela.

Todas essas considerações dizem respeito ao meio físico de interligação dos elementos da ma-

lha. No estudo das redes, levam-se em conta, ainda, as ligações lógicas possíveis entre os componentes, que se resumem em três níveis, explicados a seguir.

Nível 1. Objetiva-se o compartilhamento de periféricos tais como impressoras, plotters e modems, utilizados por mais de uma estação; devido ao baixo volume de saída desses dispositivos, não há degradação do sistema.

Nível 2. Os meios de armazenamento em massa, normalmente discos rígidos, são compartilhados pelos diversos usuários do sistema. Esse compartilhamento de periféricos requer software controlador de acesso aos arquivos de dados, para que a integridade das informações seja mantida.

Nível 3. Aqui, além da capacidade de transmitir em alta velocidade, requer-se facilidade no acesso simultâneo a arquivos, bem como a possibilidade de bloquear registros ao acesso (LOCK). Sem esse nível não é possível manter-se a integridade dos dados, eliminar a redundância de informações e consolidar dados produzidos por diferentes usuários.

Redes nacionais

Entre os fabricantes de redes para equipamentos nacionais destacam-se a Cetus, a Itautec e a NCT. A Cetus utiliza um nodo modelo CS-1000 (ou CS-1200) ou uma placa periférica (quando utilizada em compatíveis com o IBM PC). Tanto o nodo quanto a placa servem como interface física entre o meio (o par trançado) e o hardware distintos de cada máquina.

A ligação entre o nodo e o equipamento se faz via saída padrão RS232C. Os periféricos ligados a essa rede recebem também um nodo. No caso de unidades de discos rígidos, são utilizados os nodos CS-1200. A ligação de impressoras seriais também necessita de nodo.

A grande vantagem desta rede está na interligação de equipamentos diferentes, como, por exemplo, compatíveis com o IBM PC, o Apple e o TRS-80.

A rede NCT-7301 é dedicada a compatíveis com o IBM PC; sua distribuição se realiza em barramento global, que utiliza um cabo co-axial de 75 Ω para ligar os micros. Cada máquina recebe uma placa periférica que serve como interface entre o cabo co-axial e a CPU da máquina. A velocidade de transferência é de 1 Mbit por segundo, permitindo o compartilhamento de periféricos diretamente ligados aos micros (impressoras seriais ou paralelas, discos rígidos, modems etc.). Essa rede possibilita o correio eletrônico, bem como até doze unidades de discos.

A rede Itautec interliga até 32 micros I-7000. A vinculação dos elementos se faz via barramento global, por meio de cabos co-axiais, a distâncias de até 1.000 m entre o primeiro e o último micro.

Por ora, ela só permite utilização como o I-7000, embora já esteja programado um novo produto para ligar também micros de 16 bits, como o I-7000 PCxt.

São dois os tipos de estação utilizados por essa rede: micros usuários e micros servidores (onde estão conectados os periféricos que serão compartilhados pelos micros usuários). A única exigência é que exista pelo menos um servidor na rede.

O hardware da Itautec não utiliza a saída serial do micro. Uma placa de expansão instalada no equipamento permite velocidades de até 2,5 Mbits por segundo.

Fabricante	Tipo de rede	Meio físico de interligação	Compartilha	Desenvolvido para o equipamento
Cetus	Barramento	Par trançado ou co-axial	Discos, impressoras, discos rígidos, modem	Mistura qualquer equipamento
Itautec	Barramento	Cabo co-axial	Discos rígidos, impressoras	I-7000
NCT	Barramento	Cabo co-axial	Discos rígidos, impressoras, modems	Só compatíveis com IBM PC (16 bits)

BYTE A BYTE

Utilizando códigos mnemônicos e símbolos alfanuméricos, a linguagem ASSEMBLY é uma tradução direta do código de máquina, porém mais compreensível que este aos olhos humanos.

Já vimos como as linhas de um programa em BASIC são interpretadas e reduzidas a símbolos seguidos por dados ASCII. Vemos assim que o BASIC, embora considerado linguagem de alto nível, consiste apenas em seqüências de instruções; cada instrução, por sua vez, consiste numa palavra de comando seguida por dados. Os comandos são substituídos por símbolos que se encontram a apenas um nível acima dos códigos operacionais de máquina. Como as palavras de comando e seus dados (variáveis, números e strings) assemelham-se àquelas utilizadas na linguagem natural, e as instruções são separadas por números de linha ou por dois-pontos, um programa em BASIC nos parece de nível muito mais alto do que realmente é. Portanto, podemos concluir que o código de máquina precisa apenas de um pouco de tratamento visual para

Atividades de controle
O bloco de controle admite a instrução codificada (em binário), interpreta-a e induz todas as outras partes da CPU a atuar de modo adequado. Por exemplo, se uma instrução indica que o conteúdo do acumulador deve ser armazenado em determinada posição da memória, o bloco de controle coloca o endereço no bus de endereço, envia sinais instruindo a memória a armazenar dados, e coloca o conteúdo do acumulador no bus de dados.

se tornar compreensível a nossos olhos.

Esse tratamento visual é a linguagem ASSEMBLY. Nela, códigos alfabéticos mnemônicos (fáceis de memorizar) como LDA e ADC substituem os códigos de operação de um só byte, que o microprocessador entende, e utilizam-se símbolos alfanuméricos como LABEL1 e TFLAG, em vez de endereços de memória e dados numéricos. O microprocessador não compreende a linguagem ASSEMBLY; por isso, antes de executar um programa, devemos traduzi-lo para código de máquina — seja manualmente, pelo programador, seja por intermédio de um programa denominado assemblador, ou montador.

A linguagem ASSEMBLY é uma tradução direta do código de máquina. Substituindo os códigos mnemônicos por códigos de operação e os símbolos por números, convertemos um programa em código executável. Mas o ASSEMBLY revela-se muito mais compreensível aos olhos humanos do que o código de máquina, e por isso é útil na programação. Na prática, escrevemos programas em linguagem ASSEMBLY, e só nos preocupamos com as equivalências em código de máquina nos estágios finais do desenvolvimento do programa. Mas, no momento, vale a pena examinar ambas as formas, para melhor compreensão.

Sabemos que o microprocessador executa diferentes funções lógicas, mas, em essência, ele apenas manipula o conteúdo da memória. Para tanto, atua diretamente nos chips de RAM e de ROM ou trabalha com sua própria memória interna, os registradores. Estes consistem em vários bytes de memória fisicamente localizados dentro do microprocessador, e, embora possuam certas funções especiais, são indistinguíveis de quaisquer outros bytes de memória.

O acumulador

O registrador mais importante denomina-se acumulador. Está diretamente conectado à ALU (Arithmetic and Logic Unit, "unidade aritmética e lógica") e, assim, é usado com mais freqüência do que qualquer outro registrador. Para utilizá-lo, devemos introduzir as informações num processo denominado "carregar o acumulador". Usando a linguagem ASSEMBLY, dizemos que o 6502 é carregado por meio da operação LDA, e o Z80, pela operação LD A (ambas abreviações de LoaD Accumulator). Tão essencial quanto carregar o acumulador é dele extrair informações; a linguagem ASSEMBLY do 6502 utiliza para tanto a operação STA (STore the Accumulator, "armazenar o conteúdo do acumulador"). Para o Z80, no entanto, carregar e armazenar são casos diferentes de uma mesma operação: a transferência de dados. Assim, a extração de informações do acumulador também é efetuada pela operação LD A, mas num formato diferente.

Vamos supor, então, que queremos escrever um programa em linguagem ASSEMBLY para copiar o conteúdo de 1 byte de memória no byte de memória seguinte. Começamos copiando o byte $09FF no byte $0A00. Podemos expressar essa operação em linguagem ASSEMBLY assim:

6502	Z80
LDA $09FF	LD A,($09FF)
STA $0A00	LD ($0A00),A

Repare que estamos copiando o conteúdo do byte $09FF no byte $0A00, sem saber qual é o conteúdo do byte — é essencial que isso fique claro. O byte $09FF pode conter qualquer número de $00 a $FF, e tudo o que nosso programa faz é carregar esse número no acumulador e, então, transferi-lo deste para o byte $0A00. A versão da linguagem ASSEMBLY para o 6502 não deixa claro que a instrução LDA se refere ao conteúdo do byte $09FF, mas ela distingue inequivocamente entre carregar (LDA) e armazenar (STA). Já a versão Z80 não faz essa distinção em seus códigos de operação. Seu formato de instrução é sempre este:

CÓDIGO DE OPERAÇÃO DESTINO, (FONTE)

Essa versão inclui endereços de memória entre parênteses quando quer se referir ao conteúdo, o que reforça a distinção essencial entre o endereço de um byte e o que ele contém.

Uma rotina traduzida em código de máquina deve ser armazenada em algum lugar da memória para que o microprocessador a execute. Essa decisão não é familiar aos programadores de BASIC, que nunca precisam saber onde armazenar um programa — eles apenas o digitam e o executam com o comando RUN. As decisões relativas ao armazenamento já foram elaboradas pelos projetistas do sistema e implementadas pelo sistema operacional.

Um programa em código de máquina pode ser armazenado em qualquer lugar da memória, mas alguns locais são mais indicados que outros. Os lugares seguros diferem de uma máquina para outra; daí as diferentes versões de um mesmo programa, conforme o equipamento.

Códigos próprios
A memória armazena as instruções que premitem à CPU executar atividades específicas, enquanto o circuito de I/O (entrada/saída) faz a comunicação com o exterior. Cada modelo de CPU possui seus próprios códigos de instrução.

ESCHER ELETRÔNICO

M. C. Escher, artista gráfico holandês, baseava seu trabalho em formas simétricas dispostas em diversas posições. Este programa faz algo semelhante, produzindo figuras incríveis na tela do micro.

Após completar um quadrado básico com símbolos, conforme a ilustração, o computador preencherá a tela, criando seu "Escher".

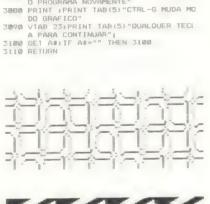
O primeiro passo é determinar o tamanho de seu quadrado básico, respondendo quando o computador perguntar. Por exemplo, se você escolher 3, terá de preencher três linhas de três posições com 0 e 1, criando um padrão de sua escolha. Em seguida, o computador perguntará o modo gráfico — alta ou baixa resolução. Você poderá então preencher a tela do monitor com seu quadrado básico a partir do canto superior esquerdo e, para cada quadrado, o computador perguntará a próxima posição. Digite: [I], para cima; [J], para a esquerda; [M], para baixo; e [K], para a direita. A rotação é comandada por: [I], 0; [J], 90; [M], 180; e [K], 270 graus. Se desejar mudar o modo gráfico pressione [CTRL]-[G]; e, para redesenhar seu quadrado básico, pressione [CTRL]-[B].

Você poderá guardar a tela gerada em disco com este comando:

```
BSAVE NOME DA TELA, A$2000, L$2000
```

E carregá-la novamente, depois, com o seguinte programa:

```
100 D$ = CHR$(4) : REM CTRL-D
110 HGR
120 PRINT D$ "BLOAD NOME DA TELA"
130 END
```

```
[illegible] REM********************************
[illegible] REM*       PARA LINHA APPLE        *
[illegible] REM*                               *
[illegible] REM*     ESCHER  ELETRONICO        *
[illegible] REM********************************
[illegible] 00000000001000000000";REM   20
  [illegible]EROS
[illegible] .20)
[illegible] 000-[illegible] 10 ,000
[illegible] [illegible]EM ** ENTRADA DO QUADRADO BASICO
  **
[illegible] [illegible]ME
[illegible] [illegible] TAMANHO DO QUADRADO
  [illegible]) 0=FIM : ";SS:IF SS=0 THEN TEXT
  :HOME :END
[illegible]0 IF SS<2 OR SS>20 THEN 120
[illegible]0 PRINT
[illegible]0 FOR I=1 TO SS
[illegible]0 HTAB 6:PRINT I":";
[illegible]0 HTAB 9:FOR L=1 TO SS:PRINT A(I,L);
  :NEXT L
[illegible]0 HTAB 8:INPUT " ";AN$
[illegible]0 LS=LEN(AN$)
[illegible] [illegible] THEN LS=SS
[illegible]0 POKE 37,PEEK(37) 2
[illegible]0 PRINT
[illegible]0 HTAB 9
[illegible]0 IF LS=0 THEN 200
[illegible]
[illegible] 0:IF MID$(AN$,L,1)="1" THEN
[illegible]
[illegible] [illegible]
[illegible] [illegible]OR L=1 TO SS:PRINT A(I,L);
[illegible]
[illegible] [illegible]RINT
[illegible]0 NEXT I
[illegible]0 RETURN
[illegible]0 REM ** ENTRADA DO MODO GRAFICO **
[illegible]0 [illegible]:PRINT
[illegible]0 INPUT "MODO GRAFICO (H L) : ";GM$
[illegible]0 IF GM$ "H" AND GM$ "L" THEN 520
[illegible]0 GM=1:HGR :HC=3
[illegible]0 IF GM$="L" THEN GM=0:GR :CO=15
[illegible]0 RETURN
[illegible]0 REM ** PREENCHE QUADRO **
[illegible]0 VTAB 21:HTAB 1:CALL -958
[illegible]0 PRINT "POSICAO (I J-K M) : ";
[illegible]0 GET A$:PRINT A$
[illegible] A$=CHR$(2) OR A$=CHR$(7) THEN RETURN
  :REM: CTRL B OU CTRL G
[illegible]0 DR=(A$="I")+2*(A$="K")+3*(A$="M")+
  [illegible]*(A$="J")
10[illegible]0 IF DR=0 THEN 1020
10[illegible]0 PRINT "ROTACAO (I-J-K-M) : ";
10[illegible]0 GET A$:PRINT A$
1090 IF A$=CHR$(2) OR A$=CHR$(7) THEN RETURN
[illegible] RT=(A$[illegible]"M")+
  [illegible]
1110 IF RT=0 THEN 1070
1120 ON DRGOSUB 1270,1290,1320,1350
1130 FOR R=1 TO SS
1140 FOR C=1 TO SS
1150 COLOR=0:HCOLOR=0
1160 ON RTGOTO 1170,1230,1210,1190
1170 IF A(R,C)<>0 THEN COLOR=CO:HCOLOR=
     HC
1180 GOTO 1240
1190 IF A(C,SS+1-R)<>0 THEN COLOR=CO:HCOLOR=
     HC
1200 GOTO 1240
1210 IF A(SS+1-R,SS+1-C)<>0 THEN COLOR=
     CO:HCOLOR=HC
1220 GOTO 1240
1230 IF A(SS+1-C,R)<>0 THEN COLOR=CO:HCOLOR=
     HC
1240 GOSUB 1370
1250 NEXT C,R
1260 GOTO 1020
1270 IF Y>=SS THEN Y=Y-SS:RETURN
1280 RETURN
1290 X=X+SS:IF GM AND X<=(280 SS) THEN
     RETURN
1300 IF X<=(40 SS) THEN RETURN
1310 X=X SS:RETURN
1320 Y=Y+SS:IF GM AND Y<=(160 SS) THEN
     RETURN
1330 IF Y<=(40 SS) THEN RETURN
1340 Y=Y-SS:RETURN
1350 IF X>=SS THEN X=X-SS:RETURN
1360 RETURN
1370 IF GM THEN 1390
1380 PLOT X+C-1,Y+R-1:RETURN
1390 HPLOT X+C-1,Y+R-1:RETURN
2000 REM ** PROGRAMA PRINCIPAL **
2010 GOSUB 100:REM ENTRADA DO QUAD. BAS
     ICO
2020 GOSUB 500:REM ENTRADA DO MODO GRAF
     ICO
2030 GOSUB 1000:REM PREENCHE QUADRO
2040 IF A$=CHR$(2) THEN 2010
2050 GOTO 2020
3000 REM  ** INSTRUCOES **
3010 HOME :PRINT :PRINT TAB(13)"ESCHER
     ELETRONICO":PRINT
3020 PRINT :PRINT TAB(5)"POSICOES:"TAB(
     27)"ROTACAO:"
3030 VTAB 9:PRINT TAB(5)"I-CIMA"TAB(27)
     "I-SEM ROTACAO"
3040 PRINT :PRINT TAB(5)"J-ESQUERDA"TAB(
     27)"J-270 GRAUS"
3050 PRINT :PRINT TAB(5)"K-DIREITA"TAB(
     27)"K-90 GRAUS"
3060 PRINT :PRINT TAB(5)"M-BAIXO"TAB(27
     )"M-180 GRAUS":PRINT
3070 PRINT :PRINT TAB(5)"CTRL-B COMECA
     O PROGRAMA NOVAMENTE"
3080 PRINT :PRINT TAB(5)"CTRL-G MUDA MO
     DO GRAFICO"
3090 VTAB 23:PRINT TAB(5)"QUALQUER TECL
     A PARA CONTINUAR";
3100 GET A$:IF A$="" THEN 3100
3110 RETURN
```

MICROTEC

A exemplo de outras empresas de alta tecnologia, a Microtec também soube conquistar um lugar de destaque no mercado. Lançando produtos eficientes, ganhou a confiança de seus usuários.

A Microtec foi fundada em 1982, tendo como objetivos a produção, o desenvolvimento e a prestação de serviços na área de sistemas, microcomputadores e periféricos.

Inicialmente a empresa desenvolveu um teclado alfanumérico de membrana flexível, o MT-200, para entrada de dados e acesso ao banco de informações. Também nesse período a empresa lançou o microcomputador MT-200, com visor de cristal líquido para controle de processos, com interface de entradas e saídas analógicas e digitais de oito a 32 canais. Esses equipamentos, bem como outros periféricos, foram devidamente aprovados pela SEI (Secretaria Especial de Informática).

Mas foi em 1983 que a Microtec desenvolveu um microcomputador altamente profissional, que, graças à sua compatibilidade com outros equipamentos, podia aproveitar grande número de programas existentes nos mercados interno e externo. Assim, o novo produto — denominado PC-2001 — compatível em hardware e software com o IBM PC, foi lançado na Feira Internacional de Informática em fins de 1983 e sua produção deu-se já no início de 1984. No primeiro semestre desse ano, a empresa fabricou e comercializou 120 unidades do PC-2001. No semestre seguinte, produziu 461 e vendeu cerca de 77 unidades por mês. E, de janeiro a junho de 1985, atingiu 1.026 unidades.

Atuando desde 1980, a Microtec trabalha em esquema de cooperação com software houses e fornecedores de produtos. Em sua nova unidade industrial, a empresa espera produzir a média mensal de quatrocentos equipamentos.

O PC-2001 foi tão bem aceito pelo mercado que a Microtec, em 1984, ampliou a configuração do equipamento, produzindo placas de suporte para expansão de memória multiusuário e comunicação com computadores de grande porte IBM e Burroughs. Em agosto desse mesmo ano, a FINAME credenciou o PC-2001, certificando seu índice de nacionalização em 87,9%.

Ocupando uma posição mais sólida no mercado, a Microtec vem aplicando grandes recursos no desenvolvimento de equipamentos, protótipos e outros projetos destinados à ampliação de periféricos, para tornar o produto mais completo. Também tem investido no potencial humano, preparando engenheiros e analistas para que possam assessorar os clientes, desde a especificação do sistema até sua utilização.

Assim, em meados de 1985, a Microtec passou a produzir placas periféricas, começando com um módulo de expansão de memória para 512 Kbytes. Posteriormente fabricou as seguintes placas: co-axial com 3274/3276 para emulação de terminais IBM 3278 e 3279; módulo de comunicação remota emulando terminais IBM 3780; módulo de comunicação Pol Select para terminais Burroughs (TDI); módulo de interface assíncrono RS232C para aplicação multiusuário; módulo de controle para Winchester de 10 a 40 Mbytes; módulo de rede local; além do terminal T-2000 para sistemas multiusuário com PC-2001/ST-2002 centrais.

Para desenvolver tantos projetos, a Microtec aumentou suas instalações. Em maio de 1985, inaugurou uma nova unidade industrial com mais de 1.200 m², local para onde foram transferidos os setores produtivos e de desenvolvimento.

Nessa nova fábrica será possível a produção de quatrocentas unidades de produtos por mês. Em matéria de recursos financeiros, a empresa tem investido uma média de 80 milhões de cruzeiros por mês em pesquisa e desenvolvimento de hardware e de software.

Ainda em 1984, a Microtec lançou o XT-2002, produto de tecnologia de vanguarda, considerado o que há de mais avançado em hardware com inteira compatibilidade com o microcomputador IBM XT.

Por outro lado, já estão concluídos os estudos para o lançamento, em 1986, do AT, outro produto de alta performance, compatível com o IBM AT.

GRAN FINALE

Com preços ainda elevados, começam a surgir no mercado sistemas avançados na área da música eletrônica. Pelas vantagens que oferecem, esses softwares tendem a uma rápida popularização.

A gravação digital vem revolucionando a música eletrônica, tornando obsoletas as gravações em fita magnética. Com esse novo método, a qualidade do som melhorou, e as técnicas empregadas atingiram novas proporções.

Na gravação digital, o som é codificado e exibido na tela do monitor de vídeo, no todo ou em amostras. Dessa forma, o usuário pode, visualizando o som, corrigir ou alterar partes da gravação.

Um dos equipamentos de amostragem mais conhecidos é o Fairlight, capaz de superar as limitações naturais dos instrumentos musicais comuns. Ele pode produzir uma amostra com entonação acima ou abaixo da capacidade do instrumento, sem lhe tirar as características sonoras naturais.

O usuário também pode, através do Fairlight, prolongar a duração da amostragem, observando as ondulações do som no vídeo. Ele consegue repetir (loop) as partes desejadas da amostra, dando continuidade ao som.

Para registrar uma música, o usuário do Fairlight pode optar entre a "Página R" ou o MCL (Music Composition Language, "linguagem de composição musical"). O primeiro exibe na tela um pentagrama, onde o usuário entra com as notas pelo teclado musical. Ocorrendo erros de tempo, o computador os corrigirá automaticamente, com base na marcação gravada em sua memória. No segundo sistema, as notas são introduzidas via teclado alfanumérico.

O Fairlight tem capacidade para gravar até oito instrumentos diferentes. Mesmo que haja uma defasagem (mínima) entre os sons, o computador pode harmonizá-los, por meio de um "relógio" interno. Dessa forma, o Fairlight permite a simulação de certos estilos musicais.

Embora esteja sendo usado por conjuntos musicais, como Culture Club e Duran Duran, e por músicos como Tomita e Jean-Michel Jarre, o Fairlight é um instrumento musical muito novo, com potencial ainda inexplorado. Apesar de sua popularidade, não é o único equipamento disponível no mercado. O Synclavier, duas vezes mais caro, possui recursos semelhantes, mas numa escala ainda maior. Nele, os dados são armazenados em discos Winchester, de 40 Mbytes,

Synclavier
Este sintetizador, da New England Digital, é considerado um dos mais avançados do mundo. Além das funções normais, possui capacidade para armazenar até 10 Mbytes de dados sonoros, o que corresponde a 1,25 milhão de notas.

Fairlight CMI
Um dos primeiros equipamentos de produção musical por computador. Seu sistema operacional é ativado através de um menu, oferecendo várias opções, desde o controle do teclado até o desenho das ondulações sonoras. Possibilita também reprodução gráfica por intermédio de impressora.

Roland MSQ-700
Conhecido como o primeiro seqüenciador compatível com a MIDI. Possui uma série de recursos MIDI e pode armazenar até 6.500 notas musicais. César Camargo Mariano emprega esse equipamento.

Yamaha KX5
O controlador remoto KX5 permite ao tecladista comandar vários sintetizadores MIDI a distância, tendo a possibilidade de executar músicas com técnica semelhante à de um guitarrista. O equipamento é carregado a tiracolo e tocado em pé.

Drumulator
Este computador rítmico, fabricado pela E-MU Systems, possui capacidade de memória de 10.088 notas. Tem fonte de geração digital e dispõe de um conjunto de peças de bateria padrão. Permite inserção de ROMs adicionais para variação de timbres

o que possibilita a gravação de um long-play inteiro sem necessidade de aparelhagem digital de 24 pistas.

Ao contrário do que muitos podem imaginar, esses equipamentos também têm suas limitações. Atualmente, todos os instrumentos de amostragem disponíveis no mercado reproduzem o som da amostra o mais próximo possível do original. Assim, toda vez que for preciso introduzir uma modificação específica, haverá necessidade da intervenção do usuário.

O Kurzweil K250 incorpora um programa para reconhecimento de padrões. Dessa forma, quando se aciona uma nota no teclado musical, surgem no vídeo várias amostras. Suas características podem ser combinadas e, assim, produzir o som individual.

A única restrição é quanto à diferenciação entre os sons dos instrumentos. O som do Kurzweil assemelha-se muito ao de um instrumento acústico, pela sensibilidade e timbre. Só que cada piano tem sua própria ressonância, mas todos os "pianos" Kurzweil têm a mesma sonoridade.

Um sistema que congrega as características do Fairlight, do Synclavier e do Kurzweil está sendo desenvolvido pela Lucasfilm, a empresa responsável pelo filme *Guerra nas estrelas*. Trata-se do ASP (Audio Signal Processor, "processador de sinal de áudio"), que pretende incorporar, num único sistema operacional, todos os tipos de recurso de som digital — apenas disponíveis nos grandes estúdios musicais.

A tecnologia digital não se restringe apenas aos dispositivos e sistemas geradores de som. Alguns modernos estúdios de gravação incluem várias unidades para o tratamento do som, como parte de seu equipamento básico. A unidade de reverberação constitui um exemplo desse processo. Nela a música é processada a fim de receber ecos, ou "reverberação", adquirindo assim um som característico.

O Quantec, uma outra unidade digital, não só fornece como também simula os diferentes tipos de reverberação que ocorrem em ambientes e espaços diferentes. Sua principal característica reside na facilidade em prolongar a reverberação para muito além dos limites naturais e físicos.

Com o Quantec é possível desenvolver a trilha sonora de um filme independente da filmagem dos atores; faz-se depois uma mixagem, sem perda de qualidade no produto final.

Valores humanos

Há um temor intrínseco por parte de muitas pessoas, músicos ou não, de que a tecnologia digital venha a prejudicar a autenticidade dos trabalhos musicais. Argumentam também que a produção musical mecanizada é artificial, desprezando os valores da espontaneidade e expressão do ser humano, para adquirir graus cada vez mais sofisticados de controle técnico.

Sabemos que essas afirmações são infundadas. Comparando o desenvolvimento da gravação di-

Notas sob controle

Nos anos 80, começa a germinar um novo tipo de músico, com todas as características da profissão, mas que não é necessariamente um profissional. Ele reúne conhecimentos musicais, de técnicas de gravação, de computação e de sintetizadores, mas não é um expert em qualquer dessas atividades.

É o sintesista, um verdadeiro "exército" de apenas um homem. Sem precisar obrigatoriamente teclar seu equipamento, ele pode criar obras grandiosas pelo computador. Instrumentistas como Vangelis, Tomita, Brian Eno, entre outros, têm impressionado multidões com seus trabalhos.

No Brasil, as experiências também já começam a dar resultados. É o caso de Lucas Kenichi Shirahata, de 25 anos, economista por profissão, que se tornou um sintesista nas horas vagas. Em seu laboratório, ele produz, entre outras coisas, jingles e spots para rádio e tevê, além de desenhar sistemas MIDI. Ele também presta consultoria para músicos. Alguns de seus clientes são Nelson Ayres, Edgard Poças (Balão Mágico), Fábio (Magazine), Yann (Metrô) e Paulo Bellinati (Pau Brasil).

O sistema utilizado por Lucas compõe-se de um núcleo gerencial formado por um microcomputador Unitron AP II, com 128 Kbytes de memória, duas unidades de discos, interface MIDI Roland MPU-401 e interlock com seqüenciador digital Roland MSQ-100.

Com esse equipamento, ele controla aproximadamente 15 000 notas musicais, com vários parâmetros, inclusive de dinâmica.

"Um usuário", diz Lucas, "pode projetar seus sistemas de um modo bastante simples e iniciar-se no mundo inexplorado da síntese musical computadorizada."

Seguem-se outros equipamentos que podem ser utilizados para desenvolver esse sistema:

Sintetizadores polifônicos: Roland Jupiter-6, Super Jupiter, JX-SP e Juno-106.

Sintetizadores monofônicos: Roland MC-202 e Roland SH-09.

Computador rítmico: Roland TR-707.

Processador de sinais: Digital Roland SDE-1000 e Aria Plate Reverb RAV-100.

gital ao domínio tecnológico da filmagem cinematográfica, observamos que, no campo da música, ainda há muito para explorar.

Outros equipamentos

Observando a trajetória do desenvolvimento da gravação digital, desde os primeiros aparelhos até os mais modernos, como o Fairlight CMI, o músico pode hesitar em decidir pela aplicação de microcomputadores em seu campo de atuação. Pode ainda recear que essa atividade não se popularize no Brasil. Mas, pelas indicações do mercado e pelo aperfeiçoamento progressivo dos sistemas, não há dúvida de que o processo é irreversível.

Como se trata de uma técnica relativamente nova, a síntese e o controle de sons por microcomputadores têm se restringido a poucas pessoas, geralmente ligadas a grandes estúdios de gravação. Mas o processo já está se desmitificando, e vem atingindo gradativamente um público maior.

No Brasil, várias pessoas, mesmo não sendo músicos profissionais, utilizam a técnica de controle de sintetizadores por computador. Para aprimorar o processo, existem à disposição diversos periféricos compatíveis com os micros mais usados para essa finalidade — Apple ou IBM PC. A interface de maior popularidade no momento, por sua versatilidade e baixo custo, é a Roland MPU-401.

O usuário que desejar introduzir um CAMS (Computer Aided Music System, "sistema musical auxiliado por computador") em seu estúdio vai precisar de um microcomputador compatível com a linha Apple, uma interface MIDI e um sintetizador. Como exemplo, o Casiotone CZ-101, um sintetizador MIDI com muita flexibilidade de aplicação, possibilitando vários timbres ao mesmo tempo.

Outra configuração existente no mercado internacional é o sintetizador modular Roland Super Quantec. Desprovido de teclado, possui um arquivo de 120 timbres para duas vozes de melodia, e um outro arquivo de 120 timbres para acordes de quatro notas. Tem também uma opção de vinte tipos de baixo, além de um computador rítmico com sistema de geração de som digital semelhante aos equipamentos mais clássicos, como o Drumulator E-MU. Apesar das características sofisticadas e do alto desempenho, seu preço equivale ao de um Fairlight.

Existem no mercado três tipos de software específicos para uso com um CAMS. O primeiro é um registrador de sons em tempo real, no qual o usuário digita as notas no sistema onde está conectado. O segundo, chamado passo a passo, possibilita visualizar notas musicais na tela e inserir informações através do teclado do computador ou do sintetizador. O terceiro software é um utilitário para programar especificamente o sintetizador digital FM DX7. Este, por sua estrutura, mostra-se um pouco mais difícil de ser programado diretamente no equipamento.

As formas do som
Estes gráficos ondulados foram criados e impressos num sistema de música Fairlight CMI. O primeiro (no alto) exemplifica um padrão de onda senoidal através de síntese digital aditiva; os outros dois apresentam amostragens de sons. As ondulações são produzidas pela gravação computadorizada da voz humana (acima) e de um trompete (abaixo). Os gráficos — tridimensionais ou topográficos — representam as variações na composição do som ao longo de um certo tempo.

PRIMEIRAS LETRAS

Primeiras palavras
Projetado para ser o primeiro contato de uma criança com computadores, My Talking Computer é muito simples de usar. As máscaras plásticas — contidas num fichário tipo espiral — são colocadas sobre um tablete sensível ao toque e seguras por uma moldura. Facilmente se inserem módulos de expansão com programas suplementares, por meio de encaixe no lado esquerdo do aparelho

My Talking Computer é um recurso de aprendizagem para crianças em idade pré-escolar. Verifique seu valor educacional e decida se, afinal, ele é ou não um computador.

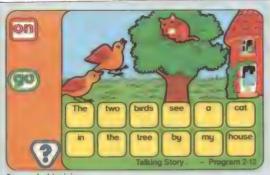
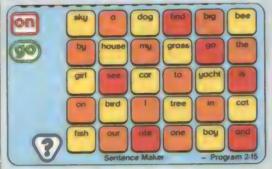
My Talking Computer, o computador falante da Microspeech, foi concebido para crianças de três anos ou mais. A companhia garante que sua máquina já é usada em escolas primárias de toda a Grã-Bretanha. O "computador" fica contido numa caixa leve, de plástico, que mede 23 x 25 cm. Máscaras plásticas com representações gráficas para vários programas são colocadas sobre um painel sensível ao toque, na parte frontal do aparelho, e seguras por uma moldura. Nesse aspecto, a máquina se assemelha ao tablete gráfico Touchmaster, embora tenha muito menos pontos de contato.

O pacote da firma inglesa inclui o relógio falante My Talking Clock, que, usado em conjunto com um dos programas incorporados, ensina as crianças a reconhecerem as horas.

Lateralmente há um encaixe, onde se introduzem cartuchos com módulos de programas. Embora o My Talking Computer seja um novo e engenhoso meio para o ensino de crianças pequenas, a aplicação dos programas residentes é limitada. Isso necessariamente diminui a vida útil da máquina, no que se refere a determinada criança. Mas os fabricantes tentam superar esse problema fornecendo um conector para expansões que possibilita o uso de ROMs adicionais.

A primeira dessas ROMs — o módulo de Expansão 1 — aparenta estar num nível apenas levemente superior ao da ROM residente. A intenção, ao que parece, é produzir uma série cuja sofisticação aumente, acompanhando o crescimento da criança. Acima do conector de expansões fica um pequeno alto-falante. A máquina funciona com cinco pilhas de 1,5 V ou ligada a uma fonte de alimentação externa.

Os programas estão residentes em um único chip de ROM, que a Microspeech afirma conter mais de 120 Kbytes de dados, embora não sejam carregados pelo computador de uma só vez. O software em ROM se divide em cinco programas mestres, cada um subdividido em vários outros, baseados no mesmo tema. O primeiro programa mestre se volta ao ensino da aritmética, com exercícios simples de reconhecimento de números, adição, subtração, multiplicação e divisão. O segundo relaciona figuras e palavras. O terceiro é o "calculador falante", que fala os números à medida que são introduzidos, e diz também o resultado. O quarto programa consiste em jogos de conversação, que testam a capacidade da criança em responder a associações como, por exemplo, "encontre o cachorro". O quinto programa, do qual já se falou, ensina a criança a ler as horas.

A criança escolhe um programa folheando as máscaras, reunidas num fichário tipo espiral, e fixando-o na moldura frontal do computador.

Contando histórias
Esta máscara destina-se a crianças de dois a doze anos. A máquina identifica o programa usado pelas posições dos quadrados ON e GO. Quando se toca um quadrado com uma palavra, esta é falada pelo computador

Formando sentenças
Este programa ensina a reconhecer palavras. Acionando a tecla [?], o computador fala uma palavra, devendo a criança pressionar o quadrado correspondente

Calculador falante
Um dos mais interessantes programas disponíveis é o calculador falante. Tocando os números e os sinais aritméticos, efetuam-se cálculos e ouvem-se os números

Aprendendo horas
O computador vem equipado com o relógio falante My Talking Clock colocado sobre o tablete de toque. O mecanismo funciona por meio de um sistema de engrenagens, situado na parte de trás do relógio. Movendo-se os ponteiros em torno do mostrador, o computador diz a hora que está sendo mostrada

Ao pressionar o quadrado ON da máscara, ligando o aparelho, uma voz feminina responde com um "Alô". A máquina solicita então que a criança pressione o quadrado GO. O computador reconhece qual programa foi selecionado porque os quadrados ON e GO estão em posições diferentes em cada máscara. Esse método de operação é particularmente útil para crianças pequenas.

Tendo aprendido que ON liga o computador e GO aciona o programa, a criança pode carregar e rodar qualquer programa sem auxílio de um adulto. As máscaras são recobertas com plástico, à prova de geléia e chocolate.

Para avaliarmos o valor educacional de um equipamento ou programa, aplicamos certos critérios. O pacote tem de ensinar o que alega e não deve pressupor qualquer conhecimento anterior do tema. Instruções escritas são inaceitáveis em pacotes do tipo "aprenda a ler". O programa deve ser fácil e divertido de usar.

Sintetizador de voz

O sintetizador de voz incorporado é a melhor coisa desse computador. De fato, faz pensar por que todas as calculadoras não têm recursos vocais incorporados. Tem suas limitações, claro. A fala é armazenada sob a forma de palavras separadas, acessadas individualmente para compor as frases usadas. A fala resultante descontínua ("de soquinhos") realmente não pode ser evitada, pois usam-se as mesmas palavras em diversos contextos. O problema mais sério — considerando que a máquina se destina a aplicações educacionais — é que não se conseguem distinguir certas palavras de outras parecidas. Mesmo repetindo algumas frases é difícil entender o que foi dito.

Pode-se agora julgar se o My Talking Computer atende ou não aos critérios definidos como essenciais num instrumento educacional. A facilidade com que se escolhem e executam os programas é um ponto positivo. O fato de o computador realmente falar constitui enorme vantagem para se ensinar uma criança a ler, evitando a necessidade de complexas instruções escritas.

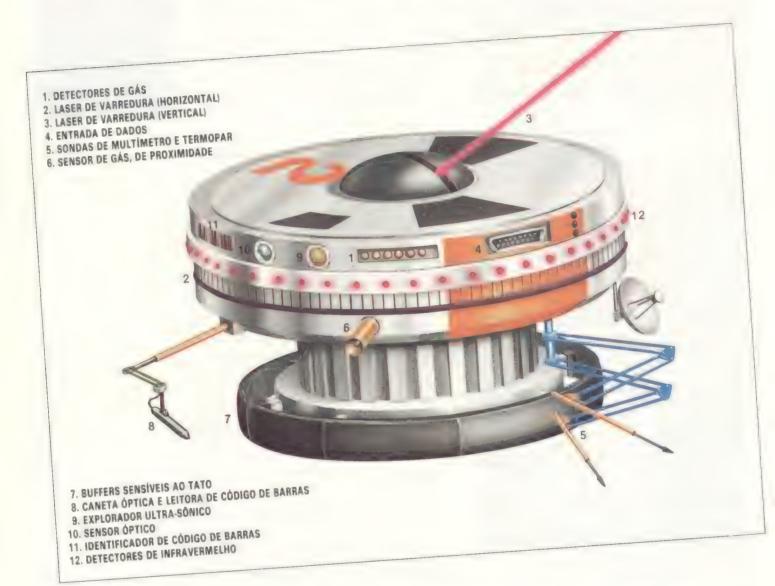
A despeito das dúvidas quanto à qualidade da fala sintetizada, a capacidade do computador em associar o som da palavra à sua forma escrita é um método de ensino direto. Bem mais direto que os oferecidos por pacotes de software que simplesmente associam a palavra a figuras — embora o My Talking Computer também faça isso. A capacidade de resposta verbal não deixa de ser um recurso vantajoso, pois assegura, também, o divertimento em seu uso.

O My Talking Computer não constitui, obviamente, um computador no sentido comum da palavra, já que não pode ser programado pelo usuário. Antes, mostra a tecnologia da microcomputação aplicada em áreas educacionais. É um brinquedo sofisticado que pode ser usado como auxiliar nas aulas.

MY TALKING COMPUTER

DIMENSÕES
250 x 230 x 70 mm

SOFTWARE
Em ROM residente no computador. Software adicional fornecido na forma de módulos de expansão em cartucho, que se encaixam num conector existente no lado esquerdo da máquina

SENSORES ROBÓTICOS

Nesta série, já examinamos como controlar os movimentos dos robôs e o projeto de "braços" e "mãos". Veremos agora de que modo eles "sentem" o que acontece à sua volta.

Para nós, humanos, o aparelho sensorial dispensa discussão. Mas, se nos faltassem os sentidos, ficaríamos completamente desamparados. Sem a visão, toparíamos com objetos; sem o tato, nem saberíamos que batemos contra algo; sem a audição, nem poderíamos ser avisados do risco de colidir com algum obstáculo. Na verdade, seríamos incapazes até de caminhar, pois os sentidos internos são indispensáveis para informar ao cérebro como o nosso corpo se movimenta.

Já vimos como um robô se movimenta, mas é preciso também dotá-lo de um sistema sensorial para que possa agir com autonomia. É realmente tentador projetar um robô dotado de todos os sentidos humanos, pois desse modo ele perceberia o mundo de forma bastante semelhante à nossa. Até hoje, porém, isso não foi possível. E, por ser tão complexo o problema da visão e da compreensão da fala por parte do robô, preferimos tratá-lo em detalhes em artigo posterior. Vamos nos limitar aqui a simples aproximações da visão e da audição das máquinas, bastante aquém do grau de complexidade dos órgãos humanos.

Para fazer um robô "ver" coisas, simplesmente lhe fornecemos um sensor de luz (em geral uma célula fotoelétrica) que produz uma tensão, a qual varia de acordo com a luz incidente sobre

Sensações robóticas
O equipamento sensorial de um robô depende de suas funções, que, quanto mais genéricas, maior número de sensores demandarão. No robô da ilustração há exemplos da maioria dos sensores possíveis e disponíveis. Mas é pouco provável que um só robô venha a receber tantos sensores

ele. Embora grosseiro, o sensor de visão dá bons resultados. Podemos, por exemplo, fazer um robô detectar e dirigir-se para uma luz brilhante, assim como ele consegue seguir uma linha. Essa propriedade permite ao robô retornar a uma fonte de suprimento de energia a fim de recarregar suas baterias. (Isso implica mais um sensor interno no robô para monitorar o estado de suas baterias, indicando-lhe quando estão fracas.)

Essa simples célula fotoelétrica capacita o robô a executar várias tarefas. Um que trabalhasse em linha de montagem seria capaz de checar a presença ou ausência de um componente detectando diferenças de brilho no local. Essa tarefa seria facilitada por uma iluminação distribuída de modo a acentuar tais diferenças. O robô pode detectar mudanças de cor pela incorporação de três células fotoelétricas, cada qual captando luzes de diferentes cores — o vermelho, o verde e o azul cobrem todo o espectro visível. Um robô assim poderia ser programado para escolher objetos vermelhos numa pilha com várias cores. Isso nos dá a impressão de um comportamento "inteligente", obtido com um sensor muito simples.

Um robô dotado de microfone "ouvirá" sinais acústicos. Não compreenderá o que ouve, mas isso pouco importa: pela repetição de um conjunto de comandos, o robô construirá um "modelo" de som para cada comando, tornando-se apto a comparar os novos sons com aqueles que registrou anteriormente. O número de comandos aos quais pode responder será limitado, mas se lhe dissermos "Siga em frente", "Pare", "Vire para a esquerda", etc., ele será capaz de seguir essas instruções.

Um robô pode ter também um sentido de tato, ainda que rudimentar. Microinterruptores incorporados estabelecem uma conexão elétrica toda vez que se aplica uma pressão sobre eles. Embora lhes falte a sensibilidade do tato humano, podem ser bastante úteis. Por exemplo, sensores táteis dispostos ao redor de um robô móvel podem capacitá-lo a reagir inteligentemente a quaisquer obstáculos: ele será capaz de se desviar e de tentar um caminho diferente. Sensores táteis incorporados à mão dos robôs fazem com que eles "saibam" quando agarrar e quanto pressionar um objeto.

Detectores de fumaça ou de gás podem ser usados para dotar o robô de um olfato rudimentar. Os de gás em geral empregam um elemento sensor (um fio de platina, por exemplo) que responde à presença de certos gases alterando a corrente elétrica que flui por ele. Os detectores de fumaça possuem duas câmaras: uma fechada, atuando como referência ou "controle", e outra aberta. Ambas contêm hélio ionizado. O número de partículas carregadas presentes na câmara aberta varia quando há fumaça dentro dela; um detector conta então o número de partículas carregadas em cada câmara, registrando a diferença entre as duas.

Até hoje não se conseguiu fornecer ao robô o sentido do paladar. Mas, empregando os métodos descritos acima, pelo menos teremos um robô que pode ver, ouvir, sentir e cheirar o suficiente para detectar um incêndio, correr em direção às labaredas, evitar os obstáculos pelo caminho e, se for dotado de um extintor de incêndio, acioná-lo sobre o fogo.

Ganho potencial

Quando limitamos um robô aos sentidos dos seres humanos, porém, perdemos muito de seu potencial. Não há por que restringi-lo ao modo de

Sensores

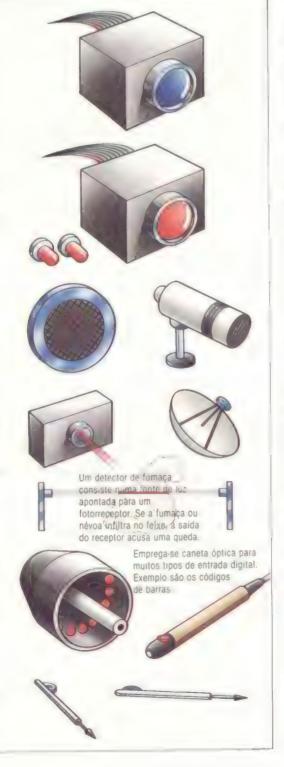
O sensor óptico é uma câmara de tevê monocromática, de baixa resolução e varredura lenta. Sua imagem, em tons cinza, contém informações suficientes ao desempenho de tarefas simples como seguir linhas ou detectar margens.

A câmara infravermelha compõe a imagem por um processo semelhante ao de tevê, mas detecta o infravermelho, e não a luz visível. O infravermelho penetra fumaça e névoa melhor que a luz, além de revelar a temperatura dos objetos.

O ultra-som é um som de alta freqüência usado para telemetria direcional. O explorador consiste num emissor de ultra-som e num microfone direcional receptor. Quando o ultra-som reflete num objeto, a textura da superfície refletora distorce a onda do eco, numa exclusiva e reconhecível "assinatura".

O explorador a laser é utilizado em goniometria e telemetria direcional de alta precisão. O feixe pode ser focalizado de maneira altamente concentrada, permitindo um exame preciso dos objetos próximos.

O detector de proximidade a gás consiste num emissor de gás e num sensor de pressão. O gás lançado na câmara provoca um aumento determinado da pressão ambiental; qualquer objeto perto da câmara afetará, de forma detectável, esse aumento.

As sondas de multímetro permitem a medição de resistência, capacitância, tensão e corrente elétrica, além de funcionarem como termopares, com os quais medimos temperaturas.

detecção que nos é peculiar. Melhor seria pensar quais sentidos poderiam ser dados a um robô e verificar se há uso prático para eles.

Um bom exemplo são os braços do robô. Vamos supor que queremos vê-lo transferir objetos de um lugar para outro. Um modo de conseguir isso seria fixando "pontos terminais" em torno do braço, para que ele percorra determinada distância máxima numa dada direção. O braço oscilaria até atingir os pontos terminais, quando então (se tudo estivesse posicionado corretamente) a mão ficaria sobre o objeto a ser apanhado. Após agarrá-lo, o braço oscilaria no sentido oposto, até que outro conjunto de pontos terminais avisasse para soltar o objeto. Este é um exemplo simples, mas demonstra que os robôs podem usar sentidos que nós humanos não temos.

Um exemplo melhor talvez seja o uso que o robô faz da visão. Nós enxergamos apenas a luz visível aos nossos olhos, mas não há razão para um robô ficar limitado a essa parte do espectro eletromagnético. Detectores de luz infravermelha adaptados em lugar das células fotoelétricas possibilitam medir a quantidade de calor gerada por um objeto. Robôs industriais podem usar esses detectores para, por exemplo, se afastar de quaisquer objetos perigosamente quentes. Mas ele também pode captar o calor do corpo humano, de modo que podemos programar nosso robô pessoal para que ele venha correndo ao nosso encontro quando chegamos em casa!

Os robôs conseguem, ainda, detectar campos magnéticos. Já falamos sobre isso quando tratamos dos robôs que seguem uma trilha no chão, mas esse é um recurso útil também nas aplicações em que eles devem diferençar entre materiais magnéticos e não magnéticos.

Sensores de proximidade não possuem equivalente humano direto, pois são meros dispositivos que detectam um objeto próximo. Com esse mesmo propósito, utilizamos uma combinação de visão e tato; mas, para o robô, um simples sensor de proximidade é suficiente. Esses dispositivos funcionam de várias maneiras. Um tipo utiliza jatos de ar, expelidos através de um bocal; qualquer objeto no campo refletirá o ar de volta, em direção ao bocal. Isso criará uma contrapressão, detectada por um transdutor, que avisará o robô da proximidade de algum obstáculo. Outro tipo depende da mudança de comportamento de um circuito elétrico com capacitância ao se aproximar de um objeto. Um "vazamento" elétrico entre o capacitor e o objeto (que terá capacitância própria) informará ao robô que há algo nas vizinhanças.

Transdutores

Também existem os sensores ultra-sônicos. Funcionam emitindo um sinal de ultra-som e captando o "eco" refletido pelo objeto próximo. O lapso de tempo entre a emissão e o recebimento do eco fornece a distância até o objeto. Esse método assemelha-se ao de orientação dos morcegos, e o princípio em que se baseia é também usado em algumas câmaras de foco automático.

Sem sentidos, sem sensações
Este braço de robô industrial limpa peças fundidas logo que saem dos moldes, quando ainda estão muito quentes para serem manipuladas por seres humanos. Sendo insensível ao calor, o robô executa o trabalho mais rápido que um operário

Mais sofisticados são os sensores a laser: dirigem um feixe de luz sobre um objeto, o qual, por sua vez, reflete-a de volta ao sensor. Comparando os dois feixes, o robô determina com espantosa precisão a distância do objeto. Pode-se usar essa técnica para grandes distâncias. Na primeira alunissagem humana, foi colocado na superfície lunar um refletor, a fim de medir-se com exatidão, através de um sensor a laser, a distância entre a Terra e o nosso satélite. A margem de erro é, no máximo, de 15 cm para uma distância de 384.400 km!

Como meio de se obterem informações táteis, os sensores de força são mais sofisticados que os microinterruptores mecânicos. Medem as variações nas propriedades elétricas de um cristal piezelétrico quando este sofre uma pressão, ou calculam a variação na condutividade elétrica de grânulos de carvão de grafita sob pressão (técnica idêntica à empregada no microfone de carbono). Podem-se usar, alternativamente, medidores de tensão para calcular forças intensas, detectando-se variações na resistência elétrica de um fio esticado.

Esses sensores chamam-se transdutores, pois tomam a medida sob uma forma (luz, pressão, som) e a convertem em outra, que representa, de algum modo, a original. Num robô controlado por computador, os transdutores convertem a medida em sinal elétrico binário (presente/ausente) ou analógico (varia segundo as mudanças da medida original). Neste último caso, o sinal é mudado em linguagem computacional por meio de um conversor analógico-digital (A/D).

Os sentidos de um robô não são tão eficientes quanto os dos humanos; mas ele possui muitos outros... a cada dia melhores!

YAMAHA CX5M

Produtora de instrumentos musicais de alta qualidade, incluindo sintetizadores eletrônicos, a Yamaha lançou o CX5M de padrão MSX — primeiro microcomputador dedicado à produção musical.

Embora siga o padrão MSX, que o capacita para aplicações como videojogos ou processamento de texto, o computador Yamaha CX5M é dirigido ao entusiasta de música eletrônica. Um teclado YK-10 ou YK-01 pode ser ligado a um conector do lado do aparelho, onde há também um par de saídas MIDI (Musical Instrument Digital Interface, "interface digital para instrumentos musicais"). Estas permitem acoplar qualquer sintetizador ou dispositivo de música eletrônica munido de interface MIDI para que funcione a partir do computador. Há também um par de microconectores para alto-falantes externos.

O computador se parece muito com outros aparelhos do tipo MSX. Possui 48 teclas tipo máquina de escrever, ladeadas por dez de controle, contendo características do padrão MSX, tais como: a tecla de apagar em retrocesso, a tecla [Graph], para exibir um conjunto de caracteres gráficos, e a [Code], que, quando pressionada juntamente com uma das alfabéticas, imprime vários caracteres em línguas estrangeiras. As cinco teclas existentes na parte superior do teclado oferecem dez funções programáveis. Quando se liga a máquina, elas assumem as funções do padrão MSX, tais como AUTO, LIST, RUN e assim por diante. O canto inferior direito do teclado abriga as quatro teclas de movimentação do cursor; acima destas, fica um conjunto de cinco teclas com outros comandos da padronização MSX — por exemplo, INSert, DELete e STOP. Na parte superior, à direita, encontra-se a entrada para cartuchos.

Arquitetura

No lado direito do CX5M há um par de saídas para joystick padrão Atari. Na parte traseira, encontram-se um conector paralelo, a interface para impressora padrão Centronics, uma saída para cassete, um conector de áudio para saída em mono, um conector para monitor de vídeo composto, um conector RF para televisor e a entrada de alimentação. As saídas MIDI e a interface para teclado musical, que conferem ao CX5M suas qualidades musicais únicas, estão contidas numa unidade que desliza, encaixando-se num conector edge sob o micro, no lado esquerdo. A razão disso é que a Yamaha pretende produzir uma série de interfaces com diferentes configurações. Quando os dispositivos estiverem disponíveis, os usuários poderão trocar as interfaces apenas removendo um parafuso e encaixando a nova interface no conector edge.

O teclado musical YK-01 (mostrado na ilustração), que alcança 3,5 oitavas, possui teclas de toque agradável, embora tecladistas profissionais possam achá-las demasiado pequenas. Utilizando o software residente, o teclado pode ser dividido, isto é, um som ("voz") programado pode ser tocado na parte grave enquanto outro som, completamente diferente, pode ser tocado com as teclas restantes. Isso significa, por exemplo, que você pode estar tocando "cordas" nas teclas agudas enquanto provê um acompanhamento de "metais" nas teclas graves. O teclado permite os modos monofônico ou polifônico (ou ambos), podendo ser tocadas até oito notas simultaneamente.

Ligando-se o micro, o monitor exibe a tela azul do padrão MSX. Acessa-se o programa de música por meio do comando CALL MUSIC, que exibirá na tela um menu contendo cinco blocos com as várias opções disponíveis. O usuário pode, então, percorrer os itens usando a tecla [Return] e alterar os parâmetros de cada opção acionando comandos do cursor. Os blocos rotulados POLY e MONO permitem ao tecladista selecionar quais dos 46 sons pré-programados disponíveis serão utilizados, e em qual dos dois modos. Esses sons variam desde instrumentos musicais convencionais (órgão, guitarra e instru-

Caixas de música
A Yamaha espera que seu CX5M venda melhor por intermédio das lojas de instrumentos musicais do que através dos revendedores de computadores. Para enfatizar sua capacidade musical, o equipamento é comercializado num pacote, que inclui o teclado YK-01 ou o YK-10. Este é ligado a uma pequena caixa com interface para música, parafusada na parte de baixo do aparelho.

Teclado
Contém todas as teclas que fazem parte do padrão MSX, inclusive cinco de funções programáveis e quatro de movimentação do cursor.

Fonte de alimentação
Por questões de espaço impostas pela interface de música, a fonte de alimentação é externa, conectada lateralmente à placa principal.

Conector edge
A caixa removível contendo a interface de música encaixa-se neste conector.

ROM do BASIC
Os 32 Kbytes de BASIC do MSX residem neste chip.

Chip do vídeo
O computador contém o chip de vídeo TMP8255, que fornece 32 sprites e dezesseis cores.

Modulador RF
Permite acoplar ao computador um aparelho de televisão comum.

Interface para impressora
O CX5M é equipado com uma saída paralela para impressora padrão Centronics.

Entrada para cartucho
Permite que o aparelho receba programas adicionais de música bem como outros cartuchos no padrão MSX.

Saídas para joysticks
Como todas as máquinas MSX, o CX5M possui duas portas analógicas no padrão Atari.

CPU
O CX5M utiliza o microprocessador Z80A de 8 bits.

Chips de RAM
Dos 48 Kbytes de RAM dentro do computador, 28 estão disponíveis para programas em BASIC.

Alguns dos cartuchos atualmente disponíveis para o CX5M. Cada um destes programas permite que o computador execute uma função diferente — desde programar o sintetizador DX7 até compor programas de música. Cada cartucho vem acompanhado de seu próprio manual. No entanto, são relativamente dispendiosos, e talvez os usuários prefiram esperar até que o preço diminua ou o software melhore.

Duas das aplicações disponíveis para o Yamaha CX5M. O programa de música é residente, sendo acessado por intermédio do comando CALL MUSIC. Movimentando-se o cursor com as teclas de funções e [Return], os parâmetros mostrados podem ser alterados acionando-se as teclas do cursor. O programa Music Composer permite ao executante escrever uma partitura musical na tela, que pode ser editada e então enviada à impressora para se obter uma cópia em papel.

mentos de percussão como vibrafone e chocalho) até uma gama de ruídos do dia-a-dia (sirene de ambulância, chuva).

Devido à natureza de alguns sons, eles não estão disponíveis quando se usa a "sustentação". A alteração dos sons nos modos POLY e MONO permite ao usuário estabelecer tons contrastantes para serem usados com o teclado dividido — ou seja, designando-se algumas teclas como monofônicas e outras como polifônicas. Naturalmente é possível usar o teclado todo em qualquer um dos modos, mas não se pode dividi-lo em duas seções POLY.

Os ruídos produzidos pelo equipamento são notáveis, mesmo quando transmitidos através de um alto-falante comum de televisão, embora alguns tenham pouca semelhança com os sons originais. Um exemplo é o "trem", que soa apenas como um órgão elétrico comum. Essa falha mostra-se freqüente entre os fabricantes de sintetizadores; mas, como em geral não se pretende uma reprodução exata de certo ruído, a maioria dos executantes ficará satisfeita com as possibilidades oferecidas.

Outro bloco, RHYTHM, proporciona acompanhamento para uma melodia. Seis diferentes padrões de ritmos estão disponíveis, utilizando-se o som de linha de baixo e de instrumentos em geral de percussão. Destes, os tons do baixo e das outras cordas são preestabelecidos pelo executante. Pode-se variar o tempo do ritmo, havendo também uma opção que permite ao executante variar a altura do som que produz o ritmo a partir do teclado de notas. Alterando-se a forma da onda sonora com o bloco LFO (Low Frequency Oscillator, "oscilador de baixa freqüência"), modulam-se os tons para a produção de diferentes sons.

O bloco BALANCE permite ajustar o volume de cada elemento rítmico de modo independente dos outros; permite, igualmente, o ajuste do volume do modo — MONO ou POLY. Assim, o volume do acompanhamento rítmico pode ser reduzido até um som de fundo, e o volume do modo POLY aumentado de maneira a realçar a melodia. Uma vez que o executante esteja satisfeito com os sons produzidos, os dados podem ser gravados em cassete e recarregados sempre que desejar. Isso o libera de designar novamente todos os parâmetros já selecionados.

Além do software residente, a Yamaha lançou uma série de cartuchos para o CX5M. Esses cartuchos incluem o Voicing Program FM, uma versão ampliada do software residente. O programa permite ao executante manipular mais eficazmente a forma de um som ajustando suas freqüências e algoritmos. Com um segundo cartucho para manipulação de sons, o computador programa o altamente qualificado sintetizador Yamaha DX7. O Music Macro FM possibilita que se programe o computador com um conjunto de comandos BASIC adicionais; já o Music Composer FM exibe uma pauta vazia e permite a entrada de uma partitura convencional e seu envio a uma impressora externa.

Embora pareça que a Yamaha oferece uma abrangente série de programas para o CX5M, fica-se com a impressão de que o software não explora plenamente as potencialidades da máquina. Em particular, os programas ligados à modulação parecem um pouco limitados — às vezes fica difícil discernir qualquer variação apreciável no som de uma nota. Esse é um fato incomum, uma vez que a Yamaha baseou o sistema operacional CX5M no mesmo do sintetizador DX7, cuja enorme variedade de sons faz o CX5M, comparado a ele, parecer "raquítico".

Além disso, embora os métodos escolhidos pela Yamaha para alterar os parâmetros dos blocos sejam adequados, às vezes parecem incômodos. Por exemplo, no uso das teclas do cursor para alterar os parâmetros enquanto o movimento do cursor é controlado pela tecla [Return] (normalmente se espera o contrário). A Yamaha já prometeu o software atualizado.

O CX5M destina-se a pessoas que desejam um microcomputador e que também se interessam por música eletrônica. Com ele, além de um sistema MIDI completo e um teclado musical, adquire-se um potencial muito maior que o proporcionado por um sistema de preço equivalente. Os não aficcionados da música eletrônica, porém, podem adquirir equipamento semelhante por menos da metade do preço.

YAMAHA CX5M

DIMENSÕES

413 x 216 x 64 mm.

CPU

Z80A.

CLOCK

3,58 MHz.

MEMÓRIA

48 Kbytes de RAM, sendo 28 Kbytes reservados para programas em BASIC.

TELA

Modo texto: 40 x 24 caracteres; modo gráfico: 256 x 192 pixels, com dezesseis cores e até 32 sprites.

INTERFACES

Impressora padrão Centronics, tevê, monitor composto, saída para áudio, saída para estéreo, duas saídas para joystick, saída para cassete, entrada para cartucho de ROM, bus para expansão, interface para teclado, conector de entrada e saída para a interface musical MIDI.

LINGUAGENS DISPONÍVEIS

BASIC, PASCAL, ASSEMBLY.

TECLADO

Estilo máquina de escrever, com 67 teclas (quatro para o cursor) mais cinco para funções programáveis. O teclado de piano é polifônico para oito notas, e cobre uma faixa de 3,5 oitavas.

DOCUMENTAÇÃO

O manual é muito falho. Enquanto dá alguns detalhes sobre como usar o software de música, sequer fornece orientação tutorial. O BASIC do MSX é apenas mencionado.

GERÊNCIA ELETRÔNICA

Usuários domésticos, empresas pequenas e mesmo de porte médio têm interesse em pacotes comerciais prontos, de baixo custo, para controle dos negócios. Este artigo analisa como funcionam.

O rótulo "pacotes comerciais" cobre uma ampla gama de aplicações, e além das diferenças de função existem outras, cruciais, de escala. A primeira grande divisão é feita entre os softwares em cassete e em disco. Estes últimos têm capacidade muito maior, e sua tecnologia de armazenamento dá aos programadores acesso a recursos rápidos de leitura e gravação. Por esses motivos, todo software comercial deveria se basear em discos — mas nem todos os usuários estão dispostos a pagar os custos adicionais de um sistema melhor.

A outra divisão de escala ocorre entre os micros de um só usuário e os de multiusuários (que acoplam vários terminais). Entretanto, mesmo os programas concebidos para sistemas multiusuários mais amplos, baseados em discos, costumam seguir uma abordagem "interativa". Isso significa que o usuário é guiado em todos os estágios da operação por uma série de indicações e mensagens na tela. As várias operações são arranjadas sob a forma de um conjunto numerado de operações em um menu. A escolha de uma função do menu principal pode levar o usuário a um menu secundário ou mesmo terciário. A tela apresentará, então, um pedido de dados, em geral informações semelhantes às utilizadas em um sistema de contabilidade manual. Essa abordagem, "amistosa ao usuário" (user-friendly), contribui para o grande sucesso dos programas de aplicações comerciais, destinados aos leigos e não a especialistas em computadores.

As três aplicações principais desses programas são o diário de vendas, o diário de compras e o diário geral. O primeiro registra as vendas feitas pelo comerciante no exercício normal de sua atividade. O diário de compras registra as aquisições necessárias para a continuidade das vendas. E o diário geral fornece, a qualquer momento, um quadro completo da situação da companhia e de seu balanço bancário.

Os programas comerciais trabalham com arquivos. Cada arquivo contém vários registros, e cada registro, vários campos. A distinção entre essas categorias é simples. Um programa de diário de vendas, por exemplo, tem um arquivo mestre de clientes com os registros de cada um deles. Os nomes, endereços e outras informações sobre cada cliente distribuem-se em campos dentro de um registro.

Além disso, é claro, um programa comercial deve operar com os dados: processar, classificar e calcular. As rotinas de entrada (que permitem introduzir os dados no sistema) e as rotinas aritméticas (que manipulam valores em campos numéricos dentro de um registro e entre registros) são exemplos de rotinas de programação necessárias.

Nos principais aplicativos, destinados ao uso em sistemas com discos, os diários de vendas e de compras registram grande número de detalhes. As vendas e compras são "lançadas" (no jargão contábil) para cada cliente e também na conta do fornecedor. A idéia é que o pacote armazene o registro mais completo possível das transações entre o comerciante e cada um de seus clientes e fornecedores.

Entretanto, mesmo os maiores sistemas baseados em discos encontram limites para a quantidade de dados que podem armazenar. Não se pode esperar que os pacotes em cassete dêem

Gerenciamento de dinheiro
Superam-se as limitações dos sistemas em cassete usando programas separados, que, se necessário, podem ser utilizados em conjunto. Este programa de faturamento deixa ao usuário apenas a tarefa de atualizar à mão seu registro de estoque. Como alternativa, é possível usar arquivos gerados por programas associados, como parte de um sistema totalmente automatizado. Uma vantagem desse processo é que se pode computadorizar a empresa passo a passo, e não de uma vez só

Livro-caixa
O software comercial em cassete é limitado quanto ao tamanho e número de programas num pacote. Como resultado, cada programa tende a se concentrar numa determinada tarefa. Este pacote tem elementos de contabilidade (registros de crédito/débito, balanços anuais etc.). Mas foi concebido somente como um auxiliar do gerenciamento do fluxo de caixa, e não fará sozinho toda a contabilidade

conta dos detalhes de tantas transações quanto os em disco — e mesmo estes podem ficar sobrecarregados. O problema-chave de qualquer usuário em potencial é o seguinte: o sistema dará conta do volume de trabalho?

Quanto maior o número de dados armazenados em um arquivo, mais tempo o computador demora para classificá-los. Assim, a abordagem normal faz com que os programas retenham informações detalhadas somente de transações pendentes, aquelas em que o dinheiro ainda não foi recebido. Os sistemas baseados nesse princípio são chamados diários de transações pendentes.

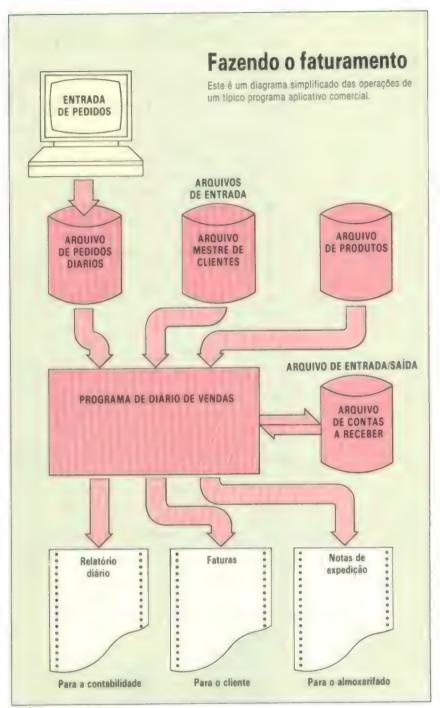
Temos uma abordagem alternativa quando o computador retém somente os valores acumulados das transações entre o comerciante e seus clientes e fornecedores — a chamada contabilidade acumulativa. Manipula menos informações, mas, em compensação, exige menos memória e é mais simples de operar. Em outras palavras, há uma permuta entre a eficiência (em termos dos detalhes armazenados) e a capacidade de processamento e de memória da máquina.

Programas em disco de diários de vendas e de compras apresentam um leque relativamente amplo de recursos. Os programas de diário de vendas, por exemplo, podem incluir uma opção de faturamento que permite ao comerciante gerar faturas e extratos para envio aos clientes. Os programas de diário de compra podem emitir cheques e pedidos de remessa (listando o que foi comprado) para encaminhamento aos fornecedores. Nada disso pode ser incorporado aos pacotes em cassetes.

Livros-caixa eletrônicos

O software comercial executa uma ampla variedade de funções. A maneira mais fácil de entendê-las consiste em considerar o fluxo de caixa de um ramo de negócios. O requisito básico de todo comerciante é determinar o faturamento e as despesas que a firma gera. O livro-caixa representa um dos métodos mais simples para se conseguir isso. Um livro-caixa manual (não computadorizado) tem suas páginas divididas em tantas colunas quantas o comerciante necessite para identificar a categoria na qual gasta ou recebe seu dinheiro. O livro-caixa também deve fazer o acompanhamento do valor dos impostos sobre cada recibo e cada pagamento, de modo que se possa pagar ou reaver do Estado a quantia correta dos impostos.

Atualiza-se o livro-caixa por dia, por semana ou por mês. Pode-se lançar cada transação ou, mais comumente, o valor total do movimento diário. A diferença entre o livro-caixa e um sistema completo de contabilidade computadorizado, que reúne os diários de vendas, de compras e geral, reside na falta de detalhamento do primeiro.

Programas comerciais ou contábeis em cassete dispõem de uma quantidade limitada de memória operacional. Não podem gravar dados em discos e, assim, liberar espaço na memória para novos dados. Por esse motivo, tendem a adotar o formato resumido do livro-caixa.

Em vez de três programas em cassete separados para os diários de vendas, compras e geral, tais programas comumente englobam todos eles. Exemplo típico seria um programa para acompanhamento semanal das entradas e saídas de um pequeno comerciante. A característica de um sistema desses é que ele consolidaria e faria o resumo dos dados introduzidos a cada semana. Para tanto, deveria totalizar as quantias recebidas e pagas no decorrer da semana, subtrair um total do outro e apresentar um relatório de lucro ou prejuízo — como faria o comerciante nos lançamentos de seu livro-caixa.

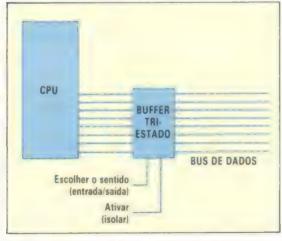
Fazendo o faturamento

Este é um diagrama simplificado das operações de um típico programa aplicativo comercial.

MANOBRAS DA MEMÓRIA

Bus de dados, bus de endereço, registro de endereços na memória, dispositivos triestado — são conceitos fundamentais na lógica da transferência de dados entre a CPU e a memória do micro.

Cada localização de memória num computador compõe-se de 8 bits. Transferem-se os dados, 8 bits por vez, ao longo de uma série de oito linhas paralelas, até a CPU, onde serão usados segundo as instruções de um programa. Também podem seguir no sentido oposto, e armazenados numa localização da memória. A instrução LDA $1234, em código de máquina, faz com que o número no endereço $1234 seja enviado através do bus de dados até a CPU. A instrução STA $1234 envia um número da CPU através do bus de dados e o armazena no endereço $1234. Assim, o bus transfere dados em ambos os sentidos.

Às vezes é importante isolar a CPU do bus de dados. Assim, cada linha deste pode estar em um de três estados (Input, Output ou Isolate). Para assegurar a passagem de um estado para outro, cada linha do bus de dados possui um pequeno circuito eletrônico, denominado dispositivo triestado. Oito desses dispositivos combinam-se num único circuito integrado. O diagrama mostra como esse circuito liga o bus de dados à CPU.

Também mostra as linhas de "ativação" e "escolha do sentido", que colocam os oito triestados no estado operacional exigido. Os circuitos também podem ser usados para conectar periféricos de entrada/saída e outros dispositivos ao bus de dados.

Quando quisermos chamar o conteúdo de uma localização, faremos referência a ela por seu endereço. Cada localização na ROM e na RAM tem seu número exclusivo de referência. Vejamos agora como, a nível de hardware, ter acesso a qualquer localização na memória de modo a executar uma transferência de dados.

A maioria dos microcomputadores possui um segundo caminho entre a CPU e a memória, chamado bus de endereço. Normalmente, este possui dezesseis linhas em vez de oito. Significa que até 65.536 endereços diferentes podem ser especificados ($2^{16} = 65.536$). Isto é, pode-se ter acesso a até 64 Kbytes de memória com um bus de endereço de 16 bits. A área de memória pode ser concebida como um conjunto de módulos, cada um contendo 256 localizações. Os 8 bits inferiores de endereços podem então ser usados para encontrar uma determinada localizaçao dentro de um dado módulo. Seleciona-se o próprio módulo por meio de alguns dos 8 bits de endereço restantes, ou de todos eles.

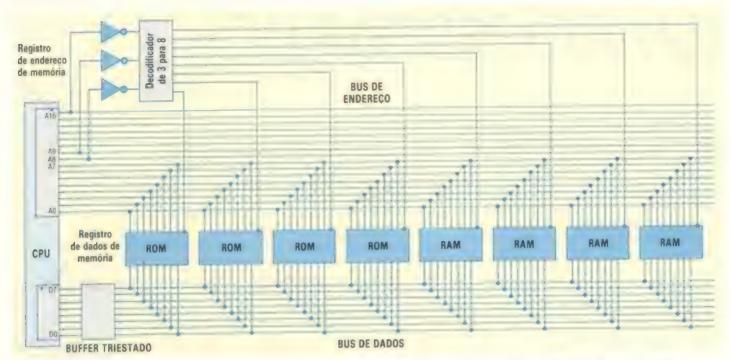
Para entender a seleção da localização, imagine um micro de 2 Kbytes de memória, igualmente dividida entre ROM e RAM. Como cada módulo de memória contém 256 localizações, o computador vai requerer oito módulos.

O endereço da localização requerida é mantido num registro especial de 16 bits na CPU, chamado MAR (Memory Address Register, "registro de endereço de memória"). Como os 8 bits inferiores do endereço selecionam uma dada localização dentro de qualquer módulo, as oito linhas mais baixas do bus de endereços podem ser conectadas a cada um dos módulos de memória. Para selecionar um determinado módulo, precisa-se apenas de mais 3 bits ($2^3 = 8$). Decodifica-se esse código de 3 bits em oito linhas de saída, uma para cada módulo.

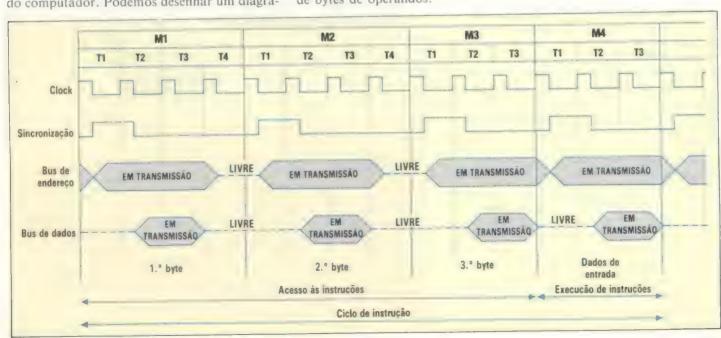
O segundo diagrama mostra como os módulos de memória ligam-se à CPU, por meio dos buses de dados e de endereço. Cada módulo tem uma única linha que chega até ele, vinda do decodificador de três (bits) para oito (linhas). Usam-se três dos bits superiores de endereço para determinar o módulo a ser selecionado. Havendo mais módulos RAM, necessita-se maior número de bits entre os 8 bits mais altos.

Código de máquina

Vamos agora examinar como a CPU executa uma instrução em código de máquina. Qualquer programa nesse código é normalmente armazenado em localizações consecutivas de memória. Uma instrução como ADD $13FF significa "somar ao acumulador o conteúdo da localização com endereço hexadecimal $13FF". Essa instrução pode requerer 3 bytes: um para conter o código binário da instrução ADD e dois para o endere-

ço de 16 bits $13FF. Digamos que seja armazenada nas localizações $1000, $1001 e $1002.

Antes de se processar a instrução, ela deve ser obtida da memória. Isso requer três acessos diferentes para levar os 3 bytes pelo bus de dados até a CPU. Ao final do ciclo de acesso, a instrução completa estará num registro especial dentro da CPU; resta decodificar a instrução e executá-la. A instrução do exemplo requer um acesso adicional à memória para ler o conteúdo da localização $13FF, a fim de somá-la ao acumulador.

Todos os fabricantes de computadores divulgam as características de seus processadores sob a forma de diagramas de tempo, que mostram a ordem dos eventos para diferentes operações do computador. Podemos desenhar um diagrama de tempo para os ciclos de acesso e execução de uma instrução em código de máquina. O controle da ordem das operações se dá a partir do pulso do clock (relógio sincronizador de operações). O gráfico abaixo mostra que, nesse sistema imaginário, a subida do pulso de sincronização ativa o bus de endereço, e a descida ativa o bus de dados. A descida do primeiro pulso de clock, em qualquer fase da operação, ou ciclo de máquina, dispara o pulso de sincronização. Os ciclos têm durações diferentes porque o processador demora mais tempo para decodificar o byte do código de operação de uma instrução do que para manipular os bytes de operandos: o código de operação deve ser decodificado imediatamente, porque especifica o número de bytes de operandos.

Acessar e executar
Uma instrução em código de máquina, consistindo em 1 byte de código de operação seguido por 2 bytes de operando, é manipulada num ciclo de instruções compreendendo fases de acesso e execução. Na fase de acesso, o bus de endereço lê as localizações da memória que contêm a instrução, e o bus de dados transporta os bytes da instrução à CPU. Na figura abaixo, os dois tipos de bus ficam ocupados durante a fase de execução porque a instrução acarreta um acesso à memória.

ESPAÇO RESERVADO

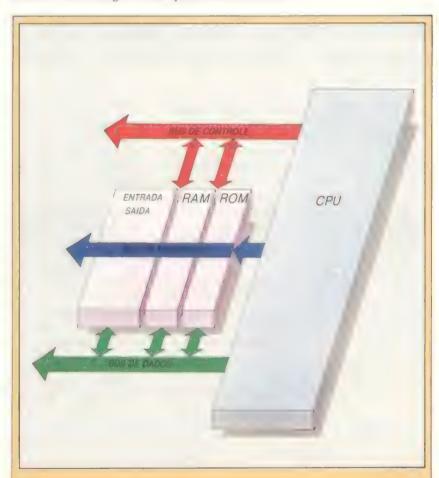
Neste artigo examinamos as formas de encontrar e reservar espaço na memória para armazenar programas em código de máquina. Veremos também como efetuar uma simples operação aritmética manipulando o acumulador.

Sabemos que qualquer programa em ASSEMBLY deve ser traduzido para código de máquina para execução. Tratando-se de um programa simples, pode-se fazer isso manualmente, o que não deixa de ter importância para a formação do programador. No entanto, à medida que os programas ficam mais longos e complexos, a conversão manual torna-se muito tediosa e propensa a erros. Nesse estágio, convém adquirir um compilador, ou programa montador, adequado ao equipamento.

Vamos examinar agora diversos aspectos da programação ASSEMBLY que podem confundir o principiante: o uso dos registradores da CPU, o local de armazenamento do programa e sua execução.

Para a CPU, a única característica relevante de qualquer byte de memória é se ele está na RAM (memória de acesso aleatório) ou na ROM (memória apenas de leitura). Os chips da ROM contêm programas e dados que requerem proteção contra regravação acidental ou deliberada e, portanto, somente podem ser lidos. Não se grava na ROM; assim, não podemos carregar nela um programa em código de máquina. Excetuando essas áreas, teoricamente nada nos impede de armazenar um programa em qualquer parte da memória. Mas certas considerações de ordem prática impedem o uso de algumas áreas.

A CPU usa alguns locais da RAM para armazenamento temporário durante suas operações. Se neles carregarmos um programa, este simplesmente será destruído pelas subseqüentes operações da CPU ou, o que é mais provável, esta lerá o código de máquina como se fosse parte de seus próprios dados. O sistema operacional também usa boa parte da RAM para armazenar seus dados de trabalho e para gerenciar o sistema todo. Assim, seria insensato ou impossível armazenar nessas áreas um programa em código de máquina, pelas mesmas razões que proíbem o uso da área de trabalho da CPU. Além disso, os programas em linguagens de alto nível podem ocupar o restante da RAM com o texto do programa e o espaço para armazenamento das variáveis. Portanto, também seria insensato carregar programas nessa área. O programador se defronta, assim, com um dilema aparentemente insolúvel.

Um recurso seria não usar o BASIC, liberando a área para o texto de programas. Mas costuma-se escrever em BASIC certas partes destes usando sub-rotinas em código de máquina somente quando o BASIC é muito lento — por exemplo, na animação de telas. Se programas em BASIC e em código de máquina coexistem na

Arquitetura básica

O sistema computacional basico compreende a CPU e a memoria. Esta se compõe de chips de ROM que contêm programas de sistemas permanentes, chips de RAM para dados e chips dedicados às operações de entrada e saida.

O fluxo dos dados e dos sinais de controle da CPU para a memoria e vice-versa se dá através de barramentos, ou buses, que são meios de transmissão semelhantes aos cabos flexíveis multivias. Eles transportam um ou mais bytes por vez e podem ser unidirecionais, como o bus de endereços, ou bidirecionais, como o bus de dados. O bus de controle transporta pelo sistema as informações que abrem e fecham as portas logicas, para direcionar o fluxo de dados. O bus de endereços transporta da CPU um endereço de 16 bits de 1 byte da memoria, onde serão armazenadas as informações que vêm atraves do bus de dados.

RAM, devemos "roubar" algum espaço e destiná-lo para o código de máquina. Podemos mudar a localização da área de texto dos programas em BASIC, ou buscar trechos que não estejam em uso, nela ou na ROM do sistema operacional.

A alteração dos limites do BASIC é muito fácil no BBC Micro, já que seus endereços ficam armazenados nas variáveis de sistema PAGE (página), TOP (alto), LOMEM (memória inferior) e HIMEM (memória superior). PAGE, por exemplo, indica o início da área de texto de programas em BASIC — normalmente no endereço $1200. Se executarmos a instrução

```
PAGE = PAGE + 500
```

o sistema operacional armazenará os programas em BASIC 500 bytes acima, deixando uma área de 500 bytes livre para os programas em código de máquina. Obtém-se o mesmo efeito em outras máquinas por meio da instrução POKE, colocando-se endereços mais altos nos indicadores do sistema. Outra possibilidade é reservar espaço baixando o endereço da parte superior da área de texto de programas em BASIC. No Sinclair Spectrum, o comando CLEAR seguido de um endereço faz exatamente isso. Ao tomarmos uma parte da área do BASIC, devemos apenas nos certificar de que restará espaço suficiente para nossos programas em BASIC.

Também se encontram pequenos blocos de espaço livre na RAM do sistema operacional, como, por exemplo, a memória intermediária (buffer) para cassete do Commodore 64. Ela

Organização interna da CPU

O processador Z80 tem uma estrutura mais complicada que o 6502. Seus registradores são usados como área de trabalho nas operações da CPU, ou para determinadas tarefas como controlar a pilha, mostrar os efeitos da última operação da CPU, ou guardar os endereços de dados ou da próxima instrução do programa.

Ambas as CPUs têm um acumulador de 8 bits conectado à ALU (Arithmetic and Logic Unit, "unidade aritmética e lógica"). O Z80 também trabalha com aritmética de 16 bits em seus registradores BC, DE e HL.

consiste em 192 bytes de RAM (de $033C a $03FB), sendo ocupada pelo sistema operacional apenas quando se utiliza o gravador cassete. Para muitos programadores, esse espaço é suficiente para armazenar suas instruções em código de máquina.

Podemos utilizar blocos de espaço ainda menores encontrados nos programas em BASIC — as linhas REM. Por exemplo, a linha de comentário

```
10 REM *************************
```

ocupa 25 bytes consecutivos com $2A, o código ASCII do "*". O sistema operacional e o interpretador BASIC nunca inspecionam esses bytes, porque para eles o comando REM significa "ignore o resto da linha". Portanto, podemos carregar uma sub-rotina em código de máquina nos bytes dos asteriscos, onde não será alterada pelo interpretador.

A grande vantagem desse método aparentemente confuso (utilizado com freqüência em programas para o micro ZX81 expandido) consiste em que, quando se salva e em seguida se carrega o programa em BASIC, a sub-rotina em código de máquina o acompanha. Com os outros métodos descritos, contudo, temos de gravar o código de máquina separadamente do programa em BASIC. A desvantagem desse método é que, ao listarmos o programa, o sistema operacional vai interpretar o código de máquina como caracteres ASCII, distorcendo a apresentação dessas linhas na tela.

Uma vez escrito um programa em linguagem ASSEMBLY, traduzido para código de máquina pela linguagem assembladora e carregado na área de RAM escolhida, passa-se à sua execução por meio dos comandos em BASIC apropriados: CALL, SYS ou USR, conforme a máquina. Segue esse comando o endereço do primeiro byte do programa. Os três comandos significam a mesma coisa para o interpretador: "execute o programa em código de máquina que começa no endereço dado e retorne para a próxima instrução em BASIC após o código de operação RET ou RTS". É semelhante, como vemos, ao comando GOSUB do BASIC.

Em BASIC, enquanto podemos escrever LET X=Y, significando "copie o conteúdo de Y diretamente em X", em linguagem ASSEMBLY temos de gravar os dados da memória na CPU, e desta para outra posição da memória. Os registradores da CPU, como mostra a ilustração, são bytes de RAM dentro da própria CPU onde se armazenam e manipulam os dados da memória. Os processadores Z80 e 6502 têm um registrador chamado acumulador, utilizado pela maioria das instruções em linguagem ASSEMBLY e responsável pelas operações.

Vamos supor que pretendemos somar os números $42 e $07 (lembre-se de que o símbolo $ indica número hexadecimal). Basta, para isso, colocar ambos os números no acumulador — a soma deles irá se "acumular" ali. Eis as instruções para a soma:

Z80		6502	
LD	A,$42	LDA	#$42
ADC	A,$07	ADC	#$07

Note que essas instruções referem-se aos números que se quer carregar e somar, e não a endereços. Na versão do 6502 o sinal # indica que se trata de um número, e não de um endereço. Assim, LDA #$65 significa "carregue no acumulador o número $65", enquanto LDA $65 significaria "carregue no acumulador o conteúdo do byte cujo endereço é $65". Também a instrução de soma, ADC (igual no Z80 e no 6502), significa "adicione esse número ao acumulador". Os números $42 e $07 chamam-se "dados imediatos", e pode-se ler LDA #$42 como "carregue o acumulador com o dado imediato $42".

Executadas essas duas instruções, a soma dos números estará armazenada no acumulador, porém ali ela será "invisível" para nós; portanto, devemos copiar o conteúdo do acumulador num byte da RAM onde possamos acessá-lo. Para isso, o programa deve terminar com uma instrução de retorno e começar com outra que coloque no estado correto um registrador da CPU. Assim seria este o programa completo:

Z80		
Código de máquina	**Linguagem ASSEMBLY**	
A7	AND	A
3E 42	LD	A $42
CE 07	ADC	A $07
32 ?? ??	LD	BYTE1 A
C9	RET	
6502		
18	CLC	
A9 42	LDA	#$42
69 07	ADC	#$07
8D ?? ??	STA	BYTE1
60	RTS	

Vamos ignorar, no momento, o significado da primeira instrução de cada versão do programa. Note, contudo, que a quarta instrução contém o símbolo mnemônico BYTE1, e não um endereço. O valor de BYTE1 varia conforme a máquina; daí usarmos aqui apenas o símbolo para, posteriormente, substituí-lo pelo endereço adequado, que deve ser o do primeiro byte após o fim do programa.

Execute o programa e verifique o resultado da soma — $49 — no byte onde estiver armazenado.

Acumulador
Principal registrador da CPU. É nele que se efetuam as operações aritméticas e lógicas, assim como a transferência de dados.

Unidade lógica e aritmética
Compreende um somador binário e portas lógicas que permitem acessar cada bit dos registradores e do bus de dados. Possibilita adições, subtrações e operações de lógica booleana.

Registrador do estado do processador
Sempre que se efetua uma operação na CPU, os bits desse registrador mostram alguns de seus efeitos — por exemplo, se o resultado é exato ou se sobra um bit transportado de uma adição.

Contador do programa
Indica o endereço da memória onde está armazenado o próximo código de operação a ser executado pela CPU.
A função USR (endereço) do BASIC carrega, nesse registrador, o endereço especificado, para que a execução prossiga a partir desse ponto.

Indicador de pilha
Contém o endereço do primeiro byte da RAM que está livre para uso da CPU. Sempre que esta introduz um novo dado na pilha, o indicador passa a apontar o próximo byte livre.

Registradores de índice e de uso geral
Os registradores de índice referem-se a endereços de memória, e os de uso geral ficam disponíveis como área de trabalho para a CPU.

EXERCÍCIOS DE ASSEMBLY

1) A direita damos a versão em ASSEMBLY de um programa simples. Passe-o para código de máquina e determine os endereços das localizações da memória.

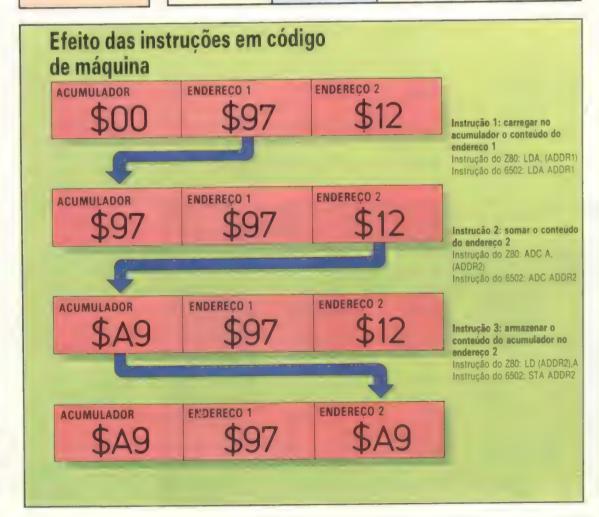
2) Que instrução está faltando no programa?

3) Qual o efeito deste programa nos registradores e na área de RAM?

4) O que significa o termo "dados imediatos"? Quais são os outros tipos de dados?

5) Se BYTE1 for considerado como um endereço, em que página da RAM ele aparecerá?

Nota. Os valores dados neste programa são apenas exemplos; se desejar executá-lo, escolha endereços e valores adequados à memória disponível.

Endereço da localização	Código de máquina	Linguagem ASSEMBLY		
		6502		
		START	EQU	$A000
		BYTE1	EQU	$45
		BYTE2	EQU	$38
			ORG	START
			LDA	#BYTE1
			CLC	
			ADC	#BYTE1
			STA	BYTE1
			ADC	#BYTE2
			STA	BYTE2
		Z80		
		START	EQU	$A000
		BYTE1	EQU	$45
		BYTE2	EQU	$38
			ORG	START
			LD	A,BYTE1
			AND	A
			ADC	A,BYTE1
			LD	(BYTE1),A
			ADC	A,BYTE2
			LD	(BYTE2),A

As instruções de transferência de dados, tais como LDA ADDR1 ou LD (ADDR2),A, copiam o conteúdo do local de origem para o local de destino, apagando os dados que ali se encontravam. O local de origem, contudo, não se altera com a operação de transferência

PROCEDA

Oferecendo serviços de bureau, equipamentos, software e desenvolvimento de sistemas, a Proceda propõe-se a acelerar o retorno do investimento do cliente.

Com o objetivo de centralizar o atendimento de serviços de processamento de dados de um conglomerado de empresas associadas foi fundada, em 1966, a Proceda Serviços Administrativos, com sede em São Paulo.

A proliferação dos recursos computadorizados e a criação progressiva de novas máquinas, cada vez menores e mais eficazes, provocaram sucessivas expansões nos equipamentos e serviços da empresa. Em 1978, acompanhando o desenvolvimento de recursos de telecomunicações no país, a Proceda já implantava diversos sistemas on line, aumentando de forma significativa sua eficiência.

Frente à constante necessidade de automação das empresas, na década de 80 a Proceda começa a expandir sua atuação a outros campos profissionais, abrangendo o mercado como um todo.

Em 1984, a Proceda ampliou sua oferta de serviços e equipamentos no ramo, estabelecendo filiais em Recife, Rio de Janeiro e Porto Alegre. No mesmo ano, considerando sua experiência no desenvolvimento e operação de sistemas complexos em diversas áreas de atividade econômica, a empresa introduziu o credenciamento de representantes para atendimento aos clientes, juntamente com sua equipe especializada, mas com planejamento e garantias centralizadas na empresa.

Atualmente a Proceda mantém filiais, além das já citadas, em Brasília, Belo Horizonte, Curitiba, Florianópolis e interior de São Paulo. Em cada pólo, a empresa conta com uma central de atendimento às necessidades dos clientes. Além da assessoria técnica, a empresa oferece treinamento para a equipe de futuros usuários durante o período de pré-instalação dos equipamentos.

Para aprimorar os conhecimentos dos clientes e desenvolver mão-de-obra especializada na operação dos equipamentos e softwares adequados a cada necessidade, a Proceda está instalando um centro de treinamento nas principais capitais do país.

Em São Paulo, um laboratório desenvolve projetos em conjunto com clientes, principalmente para a produção de softwares específicos. Esse método é aplicado quando os sistemas e equipamentos disponíveis no mercado não atendem as necessidades reais do cliente. Nesses casos, procura-se desenvolver os projetos necessários através de universidades e de fabricantes nacionais.

Para viabilizar a introdução de novos clientes nos serviços de rede de computadores e no uso adequado do bureau de serviços, a Proceda possui uma equipe de apoio especializado nessas áreas. Para as empresas que necessitam de sistemas mais complexos, a Proceda traça um plano diretor para definição de métodos, análise de alternativas, soluções e orientação ao cliente, de forma minuciosa e abrangente.

A Proceda não atinge apenas a área comercial. Possui também engenheiros especializados no levantamento dos problemas, análise e desenvolvimento de projetos de automação industrial. Dependendo do tipo de projeto e de sua complexidade, o apoio se dá a nível de assessoria para planejamento, preparo de especificações de concorrências internacionais, auditoria do integrador de sistemas, subcontratador ou integrador de sistemas.

Além desses serviços, a Proceda desenvolve sistemas para micros e minicomputadores, possuindo um complexo serviço de rede de computação para que os usuários tenham acesso a aplicações e bancos de dados. Representa equipamentos, e várias software houses nacionais e internacionais.

Serviço de rede
Seguindo a trilha do desenvolvimento das telecomunicações no país, a Proceda implantou um complexo serviço de rede, oferecendo aos usuários o acesso a bancos de dados e inúmeras aplicações

MADE IN BRAZIL

A escolha dos periféricos é o próximo passo depois de se optar pelo microcomputador mais adequado. Neste artigo, mostramos um pouco do que o Brasil já oferece nesse sentido.

Todo usuário, ao comprar seu micro, pesquisou ou procurou ajuda especializada para escolher o equipamento que melhor atendesse a suas necessidades e, ainda, acompanhasse o desenvolvimento de sua empresa. E, ao pensar nos periféricos, deve ter a mesma preocupação.

Se não forem analisados cuidadosamente e adaptados às necessidades do usuário, esses acessórios poderão se tornar tão obsoletos quanto um microcomputador mal escolhido. Além disso, na maioria das vezes, seus preços são muito elevados, não se justificando, portanto, uma compra sem real necessidade. De qualquer forma, a utilização de algum tipo de periférico é praticamente obrigatória.

Gravação de dados

Naturalmente a memória do microcomputador não é suficiente para guardar todas as informações recebidas. Assim, o usuário deve gravar essas informações em fitas ou disquetes, e liberar a memória do computador para novos registros.

Essas gravações podem ser feitas num gravador cassete simples ou nos modernos e poderosos discos rígidos tipo Winchester. A escolha depende, entre outros fatores, da quantidade de informações a arquivar.

O gravador cassete tem suas desvantagens: produz uma gravação de baixa qualidade, tem reduzida velocidade e a localização das informações na fita é difícil.

Entre os periféricos de armazenamento, os disquetes, ou discos flexíveis, são os mais populares. Nos tamanhos 5 1/4 e 8 polegadas de diâmetro, os disquetes podem armazenar entre 250.000 e 1.250.000 caracteres, numa velocidade superior à das fitas magnéticas. Para sua utilização, o usuário necessita de uma unidade de

Independência tecnológica
Graças à política de reserva de mercado — implementada pela SEI (Secretaria Especial de Informática) —, empresas nacionais puderam desenvolver a tecnologia necessária para a fabricação dos principais periféricos para micros

Monitores de vídeo

Fabricante	Modelo	Características	Tipo
Vídeo Compo	ME 9	Monocromático, 80 colunas, 9", alta resolução	FV
	CPC 14	Cromático, 80 colunas, 14"	Média resolução
	MPC 14	Monocromático, até 160 colunas, 14"	FV
	MPC 12	Monocromático, até 160 colunas, 12"	FV/FZ/FA
	MV 1	Monocromático, até 160 colunas, 12", linha Apple	FV/FZ/FA
	MV 2	Monocromático, até 160 colunas, 12", linha IBM PC	FV/FZ/FA
	MC 9	Cromático, 80 colunas, 14"	Nova versão
	CPC 14	Cromático, 80 colunas, 14"	Alta resolução
Unitron	Monitor II	Monocromático, 80 colunas, 12"	FV
Spectrum	VD 121	Monocromático, 80 colunas, 12"	FV
Unisignas	M 12	Monocromático, 80 colunas, 12"	FV
Omega	MX 200	Monocromático, 80 colunas, 12"	FV/FZ/FA
Milmar	Plus	Monocromático, 80 colunas, 12"	FV
CMA	MV 20Z	Monocromático, 80 colunas, 12"	FV
Appettomic	Apolo	Monocromático, 80 colunas, 12"	FV
CCE	MV 12	Monocromático, 80 colunas, 12"	FV

Observação: FV, fósforo verde; FZ, fósforo azul; FA, fósforo âmbar.

discos, a qual, acoplada ao microcomputador através de uma interface, fornece ou armazena as informações nos disquetes.

Outro disco com grande capacidade de armazenar informações é o tipo Winchester, rígido, que possui uma velocidade de 3.000 rpm — o equipamento da Flexidisk demora apenas 3 milissegundos para acessar a trilha. Porém, sua desvantagem ainda está no custo elevado de cada unidade.

Monitor de vídeo

Primeiro periférico necessário, pois sem ele não é possível saber o que se passa na memória do computador. Pode ser um simples aparelho de televisão adaptado ou um monitor de vídeo com alta resolução. Tanto um como o outro são úteis ao usuário, mas o monitor é mais indicado àqueles que passam muitas horas diante do vídeo. Esses equipamentos possuem alta resolução (nitidez) e uma tela de fósforo verde, âmbar ou azul que impede o reflexo da luz exterior nos olhos do operador. Dessa forma, não provoca cansaço visual nem dores de cabeça.

Mesmo para aqueles que utilizam aparelhos de televisão como monitor, a Master Sting, empresa brasileira, criou uma placa anti-reflexiva, em acrílico e poliéster, chamada Microtela. Essa máscara também impede a reflexão da luz externa e consegue resultados semelhantes aos de um monitor específico.

Os monitores de vídeo são monocromáticos (branco e preto) ou em cores, podendo ser escolhidos também pela capacidade da tela. Um monitor comum tem capacidade para apresentar, simultaneamente, até 24 linhas de 80 caracteres cada. Mas alguns, mais sofisticados, chegam até ao dobro dessa capacidade, como quatro modelos da Compo, que atingem 160 caracteres por linha. O monitor de vídeo Ouro MV20z, da CMA, também possui uma particularidade que pode ser muito útil a engenheiros e profissionais que trabalham com gráficos. Esse equipamento reduz e amplia a imagem na tela, juntando os pontos e, conseqüentemente, melhorando a visualização de gráficos.

Impressoras

De todos os periféricos utilizados por um microcomputador, a impressora é o mais procurado e o mais caro. Seu alto preço está relacionado ao desenvolvimento de sofisticados componentes.

Mesmo assim, não há usuário que resista muito tempo sem uma impressora. Ela possibilita a obtenção de cópias do trabalho realizado no vídeo, e sua escolha depende da necessidade de serviços impressos.

Alguns usuários, na ânsia de adquirir um equipamento que solucione seus problemas de textos em papel, acabam adquirindo impressoras inadequadas. Assim, os mesmos critérios adotados na compra de um microcomputador devem ser considerados na escolha da impressora.

As impressoras matriciais, que imprimem os caracteres um a um, em série, são as mais comuns e versáteis. Possuem um bloco de agulhas que tocam o papel quando acionadas, provocando a impressão. Sua velocidade varia entre 100 e 200 cps (caracteres por segundo), o que significa um trabalho relativamente rápido. Alguns modelos mais recentes, como as Elgin MT-250L e MT-440L, apresentam velocidades superiores a 250 e 400 cps, respectivamente.

As impressoras tipo margarida, ao invés de agulhas, trazem uma peça circular, dotada de "pétalas", em cujas extremidades estão os tipos (letras) para impressão. A qualidade do trabalho é mais aprimorada, ideal para cartas, mas sua velocidade mostra-se muito baixa. A impressora Anita, da Racimec, constitui bom exemplo: embora produza caracteres semelhantes aos de uma máquina de escrever elétrica, sua velocidade só vai a 18 cps.

Alguns usuários desconhecem, mas as impressoras também possuem memória. É importante, pois, conhecer a capacidade do equipamento ao adquiri-lo, visto que, quanto mais informações forem armazenadas na memória da impressora, mais rápido será o processamento.

Quando o micro se mostra mais rápido que a impressora, o usuário deve munir-se de periféricos adicionais. Os buffers, memórias adicionais que auxiliam a da impressora, armazenam muito mais informações, liberando o computador para prosseguir com o programa.

Além da capacidade de memória, dois outros fatores contribuem para a agilidade da impressora: impressão bidirecional e procura lógica. O primeiro permite que as linhas sejam impressas nos dois sentidos, utilizando também o tempo de retorno da cabeça impressora. E o segundo direciona o cursor para o ponto seguinte do trabalho, sem considerar os espaços vazios do texto.

O usuário deve analisar também o número de cópias que a impressora produz e o tipo de papel usado. Será muito importante, por exemplo, para uma empresa que emite relatórios, uma máquina que produza duas ou mais cópias. Da mesma forma, terá grande utilidade para uma firma que trabalhe com mala direta uma impressora que aceite papéis de diferentes tipos e tamanhos.

Outra forma de impressão, que oferece resultado de melhor qualidade, em que pese o custo muito alto para pequenos e médios usuários, são as máquinas de escrever eletrônicas. Acopladas a um microcomputador, conseguem um padrão bem superior ao das impressoras tipo margarida, com a vantagem de imprimir quantas cópias forem necessárias, uma a uma, em qualquer tipo de papel e com a estética escolhida no monitor de vídeo.

Um exemplo de máquina de escrever elétrica adaptável para impressão é a Edit MD, da MDA. Nessa versão, conecta-se a máquina IBM 196c ao microcomputador através de uma interface, produzindo os textos com seus próprios caracteres, na velocidade de 15 cps.

Modem

Fabricante	Modelo	Características
Elebra	DS-9601	Sincrono, 9.600 bps, linhas dedicadas, modo dúplex ou semidúplex a quatro fios, operação ponto a ponto ou multiponto
	DS-4801	Sincrono, 4.800 bps, linhas dedicadas ou comutadas, modo dúplex ou semidúplex a quatro fios, operação ponto a ponto ou multiponto
	DS-2401	Sincrono, 2.400 bps, linhas dedicadas ou comutadas, modo dúplex a quatro fios ou semidúplex a dois ou quatro fios, operação ponto a ponto ou multiponto
	DD-1921	Síncrono, banda-base, velocidade de 1.200, 2.400, 4.800, 9.600 e 19.200 bps em linhas físicas, modo dúplex a quatro fios ou semidúplex a dois ou quatro fios, operação ponto a ponto ou multiponto
	DA-1201	Assíncrono, 1.200 bps, linhas dedicadas ou comutadas, modo dúplex a quatro fios ou semidúplex a dois ou quatro fios, operação ponto a ponto ou multiponto
	DA-1031	Assincrono, 300 bps, linhas dedicadas ou comutadas, modo dúplex ou semidúplex a dois ou quatro fios, operação ponto a ponto ou multiponto
	EM-1275	Assincrono, 1.200/75 bps
CMA	217-A	Assincrono, 1.200/75 bps
Coencisa	MC-96	Sincrono, 4.800 bps, quatro fios
	MPS-48	Sincrono, 4.800/3.200 bps, dois ou quatro fios
	24 TTL-C	Sincrono, 2.400/1.200 bps, dois ou quatro fios
	MC-22	Síncrono, 1.200/600 bps, dois fios Assincrono, até 300/600 ou 1.200 bps, dois fios
	MC-23	Assincrono, até 600 ou 1.200 bps, dois ou quatro fios
	MC-16	Assincrono, até 600 ou 1.600 bps, dois ou quatro fios
	MPC-12	Assincrono, até 1.200 bps, apenas operação ponto a ponto, dois fios
	MPC-03	Assincrono, até 300 bps, dois fios
	300 TTL	Assincrono, até 300 bps, dois fios
	MAC-300	Assincrono, até 300 bps, linha comutada, operação ponto a ponto
	BBC-III	Síncrono, banda-base, 9.600/4.800/2.400/1.200 bps, linha privativa não condicionada
	BBC-II	Síncrono, banda-base, 19.200/9.600/4.800/2.400/1.200/600, quatro fios, linha privativa não condicionada

Observação: quando não mencionados nas características, os modelos operam em configuração ponto a ponto, linhas privativas ou comutadas e modo dúplex ou semidúplex.

Unidades de disco

Fabricante	Modelo	Características
Unitron	S 114" 8"	FS/DS/35 trilhas — Para compatíveis com o Apple FS/DD
Elebra	FS 00 APS 1/4" FS 00 CPS 1/4"	FS/DS/35 trilhas — Para compatíveis com o Apple FS/DD — Para compatíveis com o TRS-80
Spectrum	S 1/4"	FS/DS/35 trilhas — Usado no Apple
GCC		FS/DD

Observação: bps, bits por segundo; FS, face simples; FD, face dupla, DS, densidade simples; DD, densidade dupla.

Modem

Os telesserviços do tipo Cirandão, Videotexto e Renpac têm provocado a proliferação de um outro periférico no mercado: o modem. Sem ele, torna-se impossível o acesso de um micro a quaisquer desses serviços, via telefone.

Para a escolha de um modem, devem-se tomar os mesmos cuidados indispensáveis para a aquisição de qualquer periférico. Ele possui certas particularidades que o usuário precisa conhecer, para adquirir o equipamento mais adequado às suas necessidades.

O fator primordial consiste em verificar se o modem apresenta-se dentro das normas de utilização de linhas da Telebrás e é aprovado pelo SEI. Sem esses requisitos será impossível acessar serviços do tipo Cirandão ou Videotexto. É preciso observar também a compatibilidade do aparelho. Todas as velocidades dos modems são padronizadas pelo CCITT (Conselho Consultivo Internacional de Telefonia e Telegrafia); portanto, equipamentos com o mesmo padrão devem ser compatíveis entre si.

Para que o custo da conta telefônica não ultrapasse o orçamento da empresa, convém que o usuário analise seu tipo de serviço e então opte pelo modem com a velocidade mais indicada. Por exemplo, uma pessoa que necessita acessar os bancos de dados do Videotexto deve escolher um modem de 1.200/75 bits por segundo. Isso quer dizer que o modem recebe informações (demodula) a uma taxa de 1.200 bits por segundo e transmite (modula) a 75 bits por segundo. Sendo o número de informações a receber bem maior que o número a transmitir, sua velocidade também é maior. Isso diminui o tempo gasto na operação e, conseqüentemente, o custo.

Se, por outro lado, a operação for modem a modem, os equipamentos precisarão ter as mesmas taxas de modulação/demodulação (digamos 300/300 bits por segundo). Caso contrário, não haverá comunicação entre eles.

Para facilitar a vida daqueles que necessitam de modems com taxas de modulação/demodulação diferentes, a Elebra fabrica o EM-1275, que possui três velocidades. Ele funciona com 75/1.200, 1.200/75 e 300/300 bits por segundo. Quando se usa uma linha privada, de dois ou quatro fios, é possível, ainda, enviar e receber informações com modulação/demodulação de 1.200/1.200 bits por segundo.

Terminais

Esse periférico, embora mais específico para a área comercial, tem seu espaço garantido no mercado. Sua função é recolher e armazenar informações que serão repassadas aos computadores. Pode, entre outras operações, controlar fluxos de caixa, arquivar dados sobre inventários, funcionar como terminal de consulta a bancos de dados e de leitura de cartões magnéticos.

Exemplo desse periférico encontramos no KIM, que, além das funções normais, também roda aplicações.

Interfaces

Sem periféricos, um computador fica muito limitado ou, em alguns casos, até mesmo inoperante. Mas um micro com todos os periféricos continuará impraticável se não forem usadas as interfaces e placas de expansão adequadas para acoplá-los.

Algumas das interfaces mais populares são as de acesso a impressoras, unidades de discos, CP/M, adaptação para impressoras paralelas, expansão de memória de 16 e 32 Kbytes e módulos que mudam o padrão do vídeo para 80 colunas.

Mesmo a instalação de um modem requer uma interface e um software de comunicação específica ao banco de dados que se deseja acessar.

Quando a memória do computador se torna insuficiente ao montante de serviços da empresa, o usuário pode se valer de uma placa de expansão, que chega a elevar para 640 Kbytes a memória de um micro de 64 Kbytes.

As interfaces e placas de expansão vêm sendo cada vez mais requisitadas, pois agilizam os mais diversos tipos de computadores, além da vantagem de não pesar muito no bolso do consumidor.

Manutenção e reparos

Os equipamentos nacionais, pela facilidade de assistência técnica e de manutenção, apresentam grande vantagem sobre os importados. Isso, além de evitar a adaptação de peças, poupa o usuário de ficar com seu aparelho inativo muito tempo quando da necessidade de reparos.

Entre os periféricos de maior utilização, as impressoras merecem cuidados especiais. Mesmo antes de adquiri-las, o usuário deve estar ciente da vida útil dessas máquinas, isto é, conhecer seu período de funcionamento.

Outra informação — aparentemente óbvia, mas que, se não for observada, poderá trazer sérios aborrecimentos — é quanto ao prazo de garantia do produto. Alguns fabricantes dão prazos de garantia curtíssimos, e qualquer reparo após esse período por certo sairá do bolso do usuário.

A localização dos postos de assistência técnica também não pode ser ignorada. O usuário deve buscar sempre um equipamento que ofereça bom atendimento de manutenção e no mínimo um posto de assistência técnica em sua cidade.

Quando se trata de firmas com um volume muito grande de trabalho, é aconselhável que o contrato de manutenção seja firmado com a empresa autorizada a fornecer assistência técnica. Isso evita que se tenha de esperar "na fila" pelo atendimento, em caso de reparos.

Deve-se levar em conta, ainda, a resistência do produto, principalmente no caso de impressoras que funcionam muitas horas por dia. Não raro um aparelho parece possuir todas as características necessárias para atender às necessidades da empresa, mas tem estrutura frágil, e acaba se tornando um incômodo.

Impressoras

Fabricante	Modelo	Características	Tipo
Dismac	DP 80	80 cps/80 colunas	Matricial
Elebra	El 6010 (Mônica)	100 cps/80 colunas	Matricial
	El 6030 (Mônica Plus)	100 cps/132 colunas	Matricial/gráfica
	El 8031 (Emília I)	100 cps/132 colunas	Matricial/gráfica
	El 8035 (Emília II)	180 cps/132 colunas	Matricial/gráfica
	El 9050	200 cps/132 colunas	Matricial
	El 9051 (Alice)	250 cps/132 colunas	Matricial gráfica
Elgin	Lady	100 cps/132 colunas	Matricial
	Lady II	130 cps/132 colunas	Matricial/gráfica
	MT 440-D	300 cps/80 colunas, com software especial para a impressão em código de barras	Matricial
Ecodata	EL 8000	100 cps/80 colunas	Matricial Matricial, em versão gráfica ou não
	EL 8105	100 cps/132 colunas	Matricial/gráfica
Prológica	P 5005	150 cps/80 colunas	Matricial/gráfica
	P 750	200 cps/132 colunas	Matricial
Racimec	Ita	100 cps/132 colunas	Semigráfica
	Anita	18 cps/132 colunas	Margarida
	Caria	120 cps/40 ou 60 colunas, especial para automação comercial e bancária	
Scritta	Grafix MX80	80 cps/80 colunas	Gráfica
	Grafix MX100	100 cps/132 colunas	Gráfica
Sistema	Rima 1320	125 cps/132 colunas	Gráfica

Observação: cps, caracteres por segundo

VISÃO ROBÓTICA

Visão humana x robótica
A capacidade humana de ver baseia-se num complexo sistema de nervos e receptores, que transmitem sinais a serem processados pelo cérebro. Embora uma imagem visual consista em pontos de luz e sombra que impressionam a retina, é o cérebro quem vê. O robô faz o mesmo, mas em grau de precisão bem menor.

Transferir para as máquinas a percepção visual humana é um grande desafio da robótica. Veremos agora os princípios básicos envolvidos nessa fascinante área de pesquisa.

A visão talvez seja o mais importante dos sentidos humanos. A percepção visual é tão fundamental à nossa compreensão do mundo que, para ilustrar a dificuldade de explicar determinado assunto a quem não é capaz de entendê-lo, costumamos dizer: "É como descrever cores a um cego". Assim como a ausência da visão limita profundamente nosso conhecimento do mundo, também um robô, sem um aparelho de detecção visual, fará muito pouco. Mas como desenvolver um sistema visual robótico tão eficiente quanto o nosso? No olho humano, a íris funciona como um diafragma, controlando a quantidade de luz que nele penetra, e, na retina, uma lente focaliza a imagem. Na verdade, o olho não "vê" coisa alguma; ele é apenas um transdutor que converte um tipo de sinal em outro mais aceitável. O trabalho da visão se realiza no cérebro, com base nos sinais enviados pelos olhos.

Assim, quanto ao sistema de visão do robô, há dois problemas distintos: a construção de um "olho" que funcione como sensor e o processamento, realizado por computador, para que o robô "entenda", isto é, elabore os sinais recebidos desse sensor.

A construção de um olho para o robô é relativamente fácil. Num nível muito simples, uma célula fotoelétrica atua como uma espécie de visão, fornecendo um sinal que corresponde à iluminação geral do campo de visão. Isso se mostra útil para que o robô se dirija a uma luz brilhante, ou acompanhe uma linha branca pintada no chão sobre fundo escuro. O programa que utiliza esses dados de entrada sensoriais também é simples, uma vez que há limite para as informações recebidas, assim como para as ações que o robô executa em resposta a esses sinais.

Dificilmente, porém, chamaríamos isso de "visão". Esta seria um sistema visual capaz de construir uma imagem bidimensional completa do mundo, permitindo ao "cérebro" do robô examinar as mesmas informações processadas pelo cérebro humano e, a partir delas, chegar a uma conclusão. Uma das soluções para tal problema utiliza uma célula fotoelétrica com uma lente posicionada à sua frente. Ela varre todo o campo de visão diante do robô até construir uma imagem completa e armazená-la na memória do computador. Na prática, contudo, esse método é lento e falho.

Na maioria dos casos, o olho do robô consiste numa câmara de vídeo, seja do tipo comum usada na emissão televisiva ou especialmente projetada para a visão robótica. Alguns modelos desse último tipo empregam RAMs ópticas, que são chips especiais de memória onde o valor de cada byte é estabelecido automaticamente pela quantidade de luz que incide sobre ele. Esses dispositivos, a cada dia mais baratos, fornecem uma área de RAM com todas as informações sobre a cena observada pela visão do robô.

As informações de saída provenientes do olho do robô ficam em geral numa matriz bidimen-

O que vemos, afinal?

Qual a linha mais comprida? A ilusão de Muller-Lyer leva a pensar que a vertical da esquerda é maior que a da direita. Mas ambas têm o mesmo tamanho.

Aqui, tem-se a ilusão de que a barra superior é maior, de qualquer ângulo que se observe a figura. Na verdade, as duas têm o mesmo tamanho.

Uma figura "impossível", que mistura formas em duas dimensões. Segundo pesquisas realizadas, as pessoas não conseguem identificar esta figura como um objeto.

Embora pareçam encurvadas, as duas linhas verticais são paralelas. O olho as distorce para adequar-se à abertura dos raios.

Extraídos dos testes de Ishihara para daltonismo, estas figuras coloridas medem deficiências de percepção do vermelho e do verde. No primeiro exemplo, as pessoas com visão normal vêem o número 74, mas as daltônicas enxergam o número 21. No segundo, a visão normal nada capta, se tanto apenas um contorno esmaecido do número 2; já a daltônica vê nitidamente um 2.

sional, onde cada elemento contém um valor correspondente ao brilho da luz que incide numa determinada parte da cena observada. O número de elementos na matriz fornece a resolução da imagem, e a faixa de valores de cada elemento da matriz determina a escala de cinzas. Nos sistemas visuais, cada elemento da matriz recebe o nome de pixel, ou "elemento da imagem". Assim, uma matriz de 500 x 250 pixels, que representa os níveis de brilho alocando um byte para cada pixel, teria uma resolução horizontal de 500 pixels, vertical de 250 pixels e 256 tons de cinza, variando do preto ao branco puro. Para uma idéia do detalhe que tal imagem proporcionaria, tomemos como exemplo a imagem da televisão comum. No sistema inglês, por exemplo, há 625 linhas verticais e, portanto, resolução vertical de 625 pixels. Para uma resolução horizontal equivalente, seriam necessários cerca de 1.000 pixels, uma vez que a largura da tela é maior que a altura; os tons de cinza, representados por um único byte, gerariam 256 níveis de brilho. Um sistema robótico dotado de tal resolução forneceria uma imagem aceitável para processamento por computador.

Examinando a imagem

Para essa imagem ser "vista", o "cérebro" do robô realiza uma série preestabelecida de processos. Primeiro ajusta os tons de cinza, de modo que os pixels adjacentes com tons semelhantes sejam "uniformizados" no mesmo nível. O computador trabalha sobre a área total da imagem, calculando as médias dos níveis de cinza, eliminando as pequenas irregularidades. Feito isso, o computador examina mais uma vez a imagem, observando os pixels adjacentes com tons de cinza marcadamente diferentes; tais diferenças são então acentuadas. Pressupõem-se as características importantes da imagem delimitadas por linhas e bordas que aparecerão como mudanças bruscas no nível da escala de cinzas: o computador registra e acentua essas mudanças. Em seguida, varre de novo a imagem, buscando diferenças mais acentuadas nos tons de cinza. Utiliza-as então da mesma forma que uma pessoa liga os pontos de um quebra-cabeça para formar uma figura. O homem vale-se do fato de os pontos serem numerados. Não contando com esse auxílio, o computador simplesmente segue a rota mais provável. Ao final do processo, o robô terá obtido uma imagem uniformizada da cena, com linhas contornando os objetos mais importantes.

Mas isso significa "ver"? Na verdade, o robô apenas operou certas transformações na cena. Ele ainda não "sabe" o que está registrando.

Há duas soluções para esse problema. A primeira consiste em programar o robô com um conjunto de regras simples sobre o mundo visual. Essa é a abordagem "de baixo para cima", em que o robô começa com elementos muito simples e, a partir deles, procura elaborar o que vê num nível de compreensão mais complexo. A se-

Reconhecimento de objetos

Para reconhecer um objeto, o robô segue um processo semelhante ao do aprendizado humano: compara a imagem que vê com objetos conhecidos, até encontrar uma correspondência. Ao contrário dos seres humanos, porém, os robôs têm capacidade visual muito limitada, e armazenam um número ínfimo de padrões. Sem a profundidade da experiência humana nem sua desenvolvida capacidade de comparação, o robô não pode perceber que o objeto visto corresponde aos contornos já armazenados e conhecidos como "telefone", mesmo que em diferentes posições.

gunda solução seria programar o robô com um conjunto de objetos que têm probabilidade de surgir em seu campo de visão; ele irá então examinar a imagem, buscando a presença de algum deles. Trata-se de uma abordagem "de cima para baixo", pois o robô parte de uma idéia complexa sobre o que estaria vendo, e depois verifica se os dados de entrada visual correspondem a ela.

Reconhecimento de objetos

Exemplifiquemos a diferença entre os dois métodos com um robô que olha para uma mesa. Na abordagem de baixo para cima, ele analisa a imagem e verifica que consta de quatro partes verticais e, sobre elas, uma grande superfície horizontal. Essa constatação corresponde à informação pré-programada de que uma superfície grande pode se apoiar sobre quatro pernas, e que tal estrutura se chama mesa. Já a outra abordagem começaria com o robô perguntando "Será isso uma mesa?", visto já possuir um modelo interno de mesa, ao qual compara as informações visuais.

A abordagem de baixo para cima permite ao robô ver e compreender alguma coisa sobre objetos que jamais encontrou antes, se bem que isso exija volumosa e detalhada programação para fornecer as regras básicas sobre o que ele encontrará. A abordagem de cima para baixo, contudo, permite-lhe reconhecer apenas objetos sobre os quais já tenha informações; logo, qualquer objeto novo criará problemas. Ambos os métodos, juntos ou separados, aplicam-se aos robôs. Os seres humanos também utilizam processos semelhantes inconscientemente.

Até os dias de hoje, porém, por vários motivos, a visão do robô não atingiu a perfeição. Um dos mais importantes é a grande capacidade exigida para processar as imagens. No exemplo que demos acima, o sistema tinha 125.000 pixels, cada qual armazenado num byte — mais de 122 Kbytes de memória, portanto, para processar uma só imagem. Embora tenhamos simplificado a descrição, muitos processos realizados em cada pixel são bastante complexos em termos matemáticos. Se o robô tiver de observar os acontecimentos "em tempo real", isto é, enquanto ocorrem, receberá 25 imagens diferentes por segundo (tal como as câmaras de tevê). Assim, haveria mais de 3.050 Kbytes de dados a processar a cada segundo — o equivalente a mais de uma dúzia de disquetes!

Pode-se solucionar o problema de duas maneiras: uma é o contínuo desenvolvimento do hardware para processamento de imagens; a outra consiste em reduzir a definição da imagem e o número dos tons de cinza, a fim de utilizar o hardware existente. Dessa forma, o processamento da imagem se dá mais rapidamente, mas a qualidade resulta inferior.

Os robôs cometem vários erros ao utilizar sistemas de percepção visual. Uma solução para esses problemas estaria no desenvolvimento de sistemas em que eles "aprendam" a ver os objetos, em vez de serem programados para perceber alguns deles. Mas é possível, também, que os robôs nunca cheguem a ver de maneira adequada, pelo menos enquanto não se desenvolver um método que lhes dê um conhecimento do mundo comparável ao nosso.

SANYO MBC-550

O Sanyo MBC-550 revela-se uma opção nova e mais barata entre os micros de 16 bits. Mas as restrições de seu hardware e a ausência de software disponível ainda limitam esse bem projetado equipamento.

Diferenças entre gêmeos
O Sanyo MBC-550 é uma máquina profissional de 16 bits e preço acessível a muitos orçamentos. Roda sob o popular sistema operacional MS-DOS. Há duas versões: com apenas uma unidade de discos (MBC-550) e com duas (MBC-555). O monitor da ilustração não está incluído no preço básico.

Os computadores baseados na tecnologia de 16 bits, com exceção do QL da Sinclair, são ainda muito caros. É difícil entender a razão disso em vista da rápida queda nos preços dos processadores de 16 bits. A Sinclair demonstrou ser possível produzir com lucro máquinas nessa categoria por 40% de seu atual valor de mercado. Provavelmente o motivo esteja no perfil do cliente para esse tipo de máquina, que tende a ser o usuário profissional, razão por que os fabricantes parecem acreditar que ele é capaz de pagar um pouco mais. Isso pode ser verdade, em especial no caso dos computadores baseados nos processadores 8088 da Intel, talvez em função da política de vendas da líder no mercado, a IBM. Em vista do preço do IBM PC padrão, com uma só unidade de disco, a maioria dos fabricantes que produzem micros baseados no 8088 considera que seus equipamentos podem ganhar uma fatia do mercado, desde que sejam consideravelmente mais baratos.

Ainda assim, descarta-se uma parte do mercado, sem condições de adquirir computadores para uso profissional. Durante algum tempo, essa área do mercado era ocupada por máquinas profissionais baseadas no processador Z80, equipadas com unidades de disco que lhes permitiam rodar com CP/M. Muitos fabricantes, no entanto, perceberam o potencial dessa área negligenciada do mercado, e estão tentando produzir a preço acessível máquinas baseadas no 8088. O Sanyo MBC-550 é uma das primeiras.

Embora a Sanyo não alegue compatibilidade com o IBM PC, seu micro roda o sistema operacional MS-DOS, e pode ser encontrado tanto na versão com um único disco como com dois. Como a maioria das máquinas profissionais, o MBC-550 possui teclado separado da unidade principal. O gabinete de metal e plástico combinados tem acabamento em prata-metálico. O teclado se divide em três partes. Na extrema esquerda, estão cinco teclas de funções programáveis. Estas, usadas com a [Shift], produzem até dez funções, que variam de acordo com o aplicativo em uso. Com o BASIC, empregam-se essas teclas para comandos de uso freqüente, como LIST, LOAD e RUN.

O teclado, do tipo máquina de escrever, tem bom desenho, com todas as teclas de controle em suas posições habituais. Estas incluem a [Backspace] e a [Insert/Delete]. Há também uma [Return] muito grande, com quatro vezes o tamanho normal. À direita da barra de espaço fica uma tecla que, acionada, produz na tela caracteres gráficos. Na extrema direita encontra-se o bloco numérico tipo calculadora, que, se for pressionado, empregado em conjunto com [Number Lock], controla o cursor.

O computador propriamente dito é uma unidade metálica com o tamanho aproximado de um aparelho de videocassete, mas largo o suficiente para receber o monitor opcional da Sanyo. É fabricado, quase inteiramente, com chapas de metal, como o teclado.

Na parte frontal ficam o interruptor de força e o espaço para duas unidades de discos, embora nossa ilustração mostre um modelo de apenas uma. Essas unidades são tipo padrão e usam discos flexíveis de 5 1/4 polegadas. Cada unidade é dotada de um prendedor, que mantém o disco na posição correta e impede sua renovação durante a leitura.

Interfaces periféricas

A parte de trás do computador contém a fonte de alimentação e as interfaces periféricas. Do la-

do esquerdo estão o cabo de força, o fusível e a ligação para terra. À direita uma interface padrão Centronics possibilita a conexão de impressora paralela. Há um par de saídas para o monitor à direita. A primeira, uma saída para RGB, permite ao computador utilizar um monitor em cores; a outra é um conector de vídeo composto, especial para o monitor monocromático da Sanyo. Acima dessas saídas estão outras, de expansão, reservadas para acomodar interfaces adicionais.

A conexão Line, acima da saída para impressora e reservada para uma interface serial RS232C, torna possível a comunicação do MBC-550 com outros computadores, via modem, ou com qualquer outro dispositivo serial — uma impressora, por exemplo. Ao lado dessa saída está uma outra, na qual se pode encaixar uma interface para joysticks MBC ou compatíveis com a linha Apple. Além de possibilitar a ligação de um joystick, ela também permite controlar o computador por meio de um paddle ou periférico similar. Há também uma saída para bus externo, permitindo a conexão de outros dispositivos.

Ao contrário de muitos fabricantes, que opõem forte objeção a que usuários desmontem e mexam em seus equipamentos, a Sanyo incluiu propositadamente em seu manual instruções para colocação de todas essas interfaces periféricas, chips adicionais de RAM, uma segunda unidade de discos etc. É uma inclusão bem-vinda, embora o guia aconselhe aos usuários recorrerem a um técnico qualificado em caso de dúvida. O manual inclui, além disso, um glossário de termos de computação, uma descrição completa do BASIC da Sanyo — inclusive instruções sobre o uso do sistema em disco — e uma grande variedade de informações técnicas.

O BASIC da Sanyo, derivado da versão da Microsoft, é adequado ao equipamento, embora

Espaco para expansão
Embora o MBC 550 não seja equipado com tantas opções para periféricos como alguns dos produtos mais caros à venda, a Sanyo preparou a máquina para expansão. Há conexões adicionais para joysticks e dispositivos seriais. Acompanham um manual do usuário, instruções para o acoplamento de periféricos.

Interface para impressora
Conecta qualquer impressora paralela compatível com o padrão Centronics.

Chip para interface de periféricos
O chip 8255A controla os vários periféricos que podem ser conectados ao computador.

Chip para vídeo
Permite que qualquer parte dos 256 Kbytes de RAM seja usada para memória de vídeo.

Alto-falante
Os sons do computador são canalizados para este alto-falante embutido.

Abertura para unidade de discos
Espaço reservado para uma segunda unidade de discos.

Unidade de discos
Unidade padrão para discos flexíveis de 5 1/4", usada pela maioria das máquinas que operam com o MS-DOS.

Co-processador
Espaço reservado para inclusão de um co-processador aritmético 8087.

CPU
O MBC-550 usa o chip 8088 da Intel como unidade central de processamento.

Chips de RAM
O MBC-550 na versão básica possui 128 Kbytes de RAM. Observe, entretanto, que há vários conectores vazios para acréscimo de RAM adicional.

Alimentação
O MBC-550 da Sanyo possui transformador próprio.

pareça um pouco lento no processamento de funções aritméticas. Já se comentou muitas vezes a lentidão do chip 8088 em operações matemáticas (daí a popularidade do 8087 como co-processador aritmético), mas o MBC-550 parece ainda mais lento que outras máquinas baseadas no 8088. O desenho de linhas também é vagaroso, pois o BASIC da Sanyo não possui o comando DRAW; por isso se executa esse desenho pelo comando PSET, que simplesmente dá a um determinado pixel uma certa cor. Assim, o desenho de uma linha envolve o uso de um loop.

Além dos comandos do tipo Microsoft, como MID$, LLIST e CIRCLE, há também comandos para definir e visualizar janelas na tela. A instrução condicional WHILE-WEND incentiva a programação estruturada. Muitos comandos do BASIC da Sanyo podem ser introduzidos por meio de duas ou três teclas combinadas com a [Control]. Por exemplo, introduz-se o comando DIM pressionando [CTRL], [Shift] e [D] simultaneamente; e PRINT, pressionando [CTRL] e [P]. Embora qualquer esforço para simplificar a entrada de comandos seja bem recebido e muitas abreviaturas tenham uma relação com os comandos originais, dificilmente imaginaríamos alguém tentando decorar os quarenta comandos disponíveis para programar em BASIC. É provável que tente memorizar algumas das versões abreviadas mais comumente usadas.

Programas incompatíveis

Acompanham o Sanyo MBC-550 um disco de sistema MS-DOS, que também contém o BASIC, o conhecido processador de texto WordStar, e a planilha financeira CalcStar. Um manual com descrições completas sobre seu uso vem junto a esses pacotes.

Seria de esperar que o MBC-550 — baseado no chip 8088 e operando sob o controle do MS-DOS — não tivesse muita dificuldade em executar a grande quantidade de software disponível para o IBM PC. Mas o fato é que, devido a limitações de hardware, nenhum dos programas compatíveis com o equipamento IBM testados no MBC-550 funcionou.

Sem dúvida, existe mercado para computadores profissionais controlados por MS-DOS de baixo custo que rodam software projetado para o IBM PC. A questão não é saber se essas máquinas virão, mas quando. O MBC-550 da Sanyo foi uma primeira tentativa de quebrar a barreira do alto preço. Infelizmente, a falta de compatibilidade significa que ele permanecerá isolado.

Para o pequeno empresário, que precisa de um computador apenas para processar textos e planilhas financeiras, o MBC-550 parece ser um bom investimento. Entretanto, quem pretende ter acesso a uma variedade de software mais ampla que a disponível para o Sanyo, precisa conformar-se a uma despesa maior; ou, então, deve esperar um pouco mais.

SANYO MBC-550

Intel 8088 de 16 bits.

3.6 MHz

Modo texto: 80 x 25 caracteres: modo de alta resolução: 640 x 200 pixels, com até oito cores.

Tipo máquina de escrever, mais cinco teclas para dez funções programáveis e bloco numérico de calculadora com dezenove teclas.

Saida paralela padrão Centronics, com opções para uma interface RS232C, e saida para joystick compatível com a linha Apple

BASIC da Sanyo.

O manual é muito bom, com uma explicação completa dos comandos do BASIC, embora não inclua orientação sobre o uso. Há também uma introdução ao MS-DOS, ao WordStar e ao CalcStar. Ao contrário de outras máquinas profissionais, o guia inclui grande quantidade de informações técnicas, inclusive instruções que orientam o usuário na colocação de outras placas de expansão.

FIM DAS PILHAS

Liberte sua secretária das pilhas de envelopes: um sistema eficaz de mala direta só requer uma impressora, um pacote de processamento de textos e um programa de banco de dados.

Uma das tarefas mais cansativas nos escritórios é o endereçamento manual de enormes pilhas de envelopes. Um sistema computadorizado elimina esse trabalho tedioso e oferece recursos que vão muito além da simples impressão de etiquetas de endereços.

De fato, se o único objetivo fosse endereçar envelopes automaticamente, um software de banco de dados bastaria. Cada registro de nome e endereço seria introduzido campo a campo e, quando o sistema estivesse montado, poder-se-ia utilizá-lo continuamente para gerar tantas etiquetas quantas fossem necessárias.

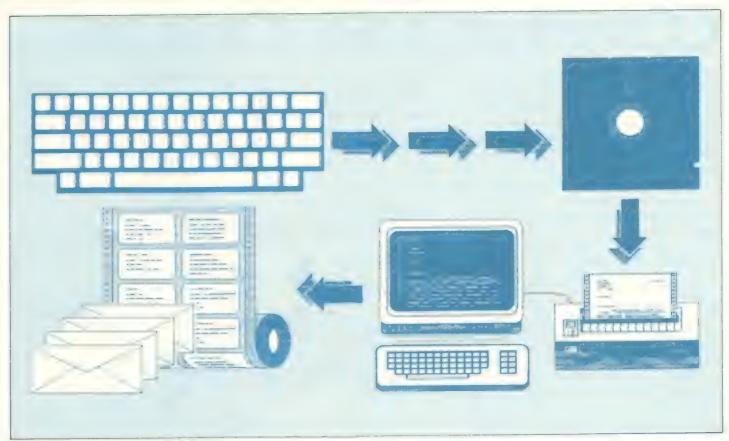
A maioria dos pacotes de banco de dados permite selecionar as etiquetas que serão impressas de acordo com certos critérios: por exemplo, todos os endereços que tenham São Paulo no campo reservado à cidade. Isso daria conta da escolha e endereçamento das etiquetas, mas deixaria ao usuário a segunda metade de um trabalho cansativo. Se o objetivo fosse enviar uma carta padronizada a um determinado conjunto de clientes, seria necessário datilografar o endereço em cada carta. Além disso, o texto permaneceria impessoal. Não haveria sentido em deixar espaços no texto para posterior preenchimento com o nome de cada cliente, pois os nomes têm tamanhos diferentes. Seria necessário um espaço suficiente para o nome mais extenso da lista, e a carta "personalizada" pareceria falsa. Ou seja, uma personalização efetiva implicaria datilografar tantas cartas quantos nomes houvesse na lista de mala direta.

Num sistema ideal, o usuário poderia primeiro compor uma carta padrão usando todos os recursos de um processador de textos. A seguir faria o computador selecionar elementos relevantes de cada cliente, a partir de um banco de dados, e os acrescentaria à carta.

Há vários meios de se fazer isso. Um dos mais simples é adotado por dois pacotes em disco produzidos pela firma inglesa Acorn: o processador de textos Memoplan e o gerenciador de banco de dados Fileplan. Ambos integram um conjunto de softwares oferecido pela Acorn aos compradores do Z80, seu segundo processador para o microcomputador BBC Modelo B.

É indispensável que os dois programas sejam usados em conjunto. O arquivo de nomes e endereços deve ser montado no Fileplan, e desenvolve-se a carta (ou formulário) padrão com o Memoplan. Cada campo do arquivo de nomes e endereços é numerado e, nos espaços em branco da carta padrão, o usuário digita o número do campo cujo conteúdo deve ser transferido para lá. O computador percorre seqüencialmente o arquivo de mala direta e insere na carta padrão o conteúdo dos campos numerados.

Ao mesmo tempo, o processador de textos corrige os espaços entre as palavras: o destinatário recebe uma carta que parece ter sido escrita especialmente para ele. O Memoplan também permite que a pessoa ajuste a carta padrão para incluir um lembrete, facilitando a identificação dos campos. Por exemplo, 2(SOBRENOME) indica que o segundo campo contém o sobrenome do destinatário.

Os pacotes de mala direta mais conhecidos, como o Mailmerge, da Micropro, ou o Mailing List, da Peachtree, destinam-se a microcomputadores profissionais, como o IBM PC, e incluem várias características sofisticadas. Podem criar

grande número de diferentes arquivos de endereços, e apresentam recursos de seleção e pesquisa que permitem a emissão seletiva de partes de uma lista. A secretária de um clube desportivo que utilizasse um sistema desses não teria problemas em enviar correspondência a todos os sócios que já tivessem participado de competições e estivessem com as mensalidades em dia.

A pesquisa e a seleção baseiam-se na indexação dos registros e classificação por chaves, técnicas padrão de bancos de dados. Efetuam-se, a seguir, testes lógicos de certos campos no registro, para verificar se atendem às condições específicas para inclusão na lista.

Esses pacotes avançados também possuem inúmeros recursos de formatação. Isso é particularmente útil, pois os nomes e endereços gerados podem ser ajustados ao tamanho e ao formato das etiquetas. Com o sistema Peachtree, por exemplo, a escolha no menu principal da opção LABEL FORMAT (formato da etiqueta) produz na tela a imagem de um retângulo juntamente com uma relação de todos os campos do registro. Isso dá ao usuário uma representação visual da etiqueta que está sendo criada. Também existem recursos para informar a impressora de que, por exemplo, as etiquetas vêm de três em três, de modo a imprimir três etiquetas a cada passagem da cabeça de impressão. Os programas de mala direta têm maior utilidade no processamento de listas grandes. Se houver apenas um punhado de etiquetas a preencher, será mais fácil datilografá-las uma a uma — a menos que as informações já estejam no computador.

Um exemplo de pacote concebido para o micro BBC Modelo B é o fornecido pela GCC, uma companhia inglesa de software. Durante algum tempo, a GCC comercializou o Starbase, um banco de dados em ROM, com os recursos necessários para mala direta. Hoje, porém, já existe uma versão do Starbase, em ROM de 16 Kbytes, que associa os recursos de mala direta ao processador de textos Wordwise.

Os recursos incluem a personalização de cartas padrão e a formatação de etiquetas. Podem-se enviar comandos à impressora enquanto se processa a formatação das etiquetas. Assim, selecionam-se diferentes tipos de letras conforme os recursos da impressora.

Uma característica bastante útil do Starbase, enquanto pacote de mala direta, é sua capacidade de efetuar operações aritméticas em campos do arquivo de endereços. Dessa forma, ele cria extratos e faturas personalizados, realizando os cálculos necessários.

Os usuários do Commodore 64 têm boas alternativas para pacotes de mala direta. Um exemplo é o Visawrite, produzido pela Visa Software. Ao contrário do Starbase, o Visawrite trabalha com cassete ou disco, e pode ser usado em conjunto com qualquer banco de dados que crie um arquivo seqüencial. Quando usado como banco de dados, armazena até quinhentos nomes e endereços. Utilizado assim, sozinho, o Visawrite não tem recursos de pesquisa seletiva. Ou seja, o usuário deve percorrer a lista de nomes e endereços e marcar os que serão incluídos na emissão de uma determinada correspondência.

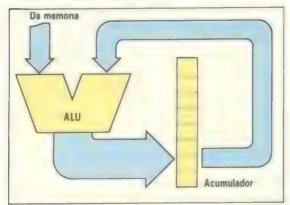
Endereçamento em cadeia
O processador de textos fornece uma carta padrão com espaços em branco nos quais se acrescentam a data, o nome e o endereço do cliente e outras informações do gênero, obtidas dos registros do banco de dados. Uma vez preparadas as cartas, os endereços são impressos em etiquetas adesivas.

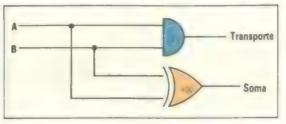
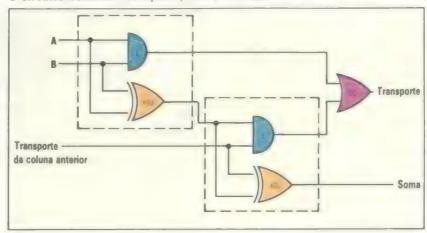
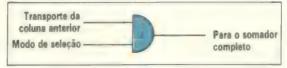
COMBINAÇÕES LÓGICAS

A CPU controla a transferência dos dados, supervisiona a execução de instalações do programa e executa cálculos aritméticos e lógicos. Vejamos, agora, os aspectos lógicos da CPU, em especial a função da ALU.

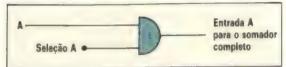
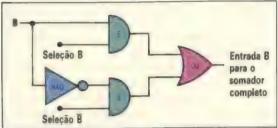
As funções que a ALU (Arithmetic and Logic Unit, "unidade lógica e aritmética") desempenha recaem em duas categorias: funções aritméticas — adição, subtração e aumento (soma de uma unidade a um número) — e três funções lógicas: OU exclusivo (XOU), OU inclusivo e E.

Algumas dessas funções, como a soma, exigem dois operandos (isto é, números sobre os quais se efetua a operação). Outras, como o aumento, exigem um único operando. Nesse caso, obtém-se o operando necessário com um registrador especial, dentro da CPU, chamado acumulador. Quando são necessários dois operandos, conseguimos o segundo deles da memória principal. Os dois números prosseguem então através do sistema de circuitos da ALU, e a operação selecionada se realiza. O resultado é recolocado no acumulador.

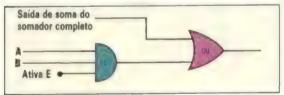
O diagrama a seguir mostra o fluxo de dados através do chip da ALU:

Os números no acumulador e na memória possuem um tamanho de palavra de 8 bits. Os bits que constituem os números passam paralelamente pela ALU, e a operação se realiza em todos os 8 bits ao mesmo tempo. Para descrever o sistema de circuitos da ALU, tomamos um dos 8 bits e projetamos um circuito que permita desempenhar todas as seis funções diferentes. Denominamos A o bit do primeiro operando e B o do segundo.

A base para o estágio de um único bit da ALU aqui examinada é o somador completo, cujo circuito, já projetado nesta seção, usava dois circuitos somadores parciais — portas E, OU e NÃO. Mas esses somadores parciais podem utilizar uma porta XOU para simplificar o sistema de circuitos.

Os dois somadores parciais se unem para formar o circuito somador completo, desta forma:

Pelo acréscimo de pequenos circuitos ao somador completo, é possível demonstrar como as portas deste podem ser adaptadas para executar as outras funções da ALU. Para tanto, estabelecemos uma série de sinais de controle. O primeiro deles — o sinal de seleção de modo — liga ou desliga a entrada de "transporte da coluna anterior", usada durante as operações aritméticas, mas é desnecessário nas operações lógicas. Isso é obtido por meio de uma porta E.

Para cálculos aritméticos, em que a entrada de transporte é necessária, coloca-se o sinal de seleção de modo em zero. De maneira semelhante, podemos acrescentar portas E às duas outras entradas. Isso nos permite selecionar ou o bit A, ou o bit B, ou ambos.

Durante o processo de subtração por meio de adições de complementos a dois, deve-se calcular tal complemento do número a ser subtraído. Isso exige a mudança de todos os 1 para 0 e vice-versa. Se nosso circuito somador for usado para subtração, então teremos de acrescentar algum sistema de circuito que nos permita selecionar a negação do bit B. Pode-se fazer isso fornecendo a entrada B por meio de uma porta NÃO e incorporando o sinal de seleção mediante outra porta E. O sistema final de cir-

cuitos para as entradas A e B, juntamente com o sinal de controle, é o seguinte:

Com esses quatro sinais de controle, realizamos cada uma das funções aritméticas. A tabela a seguir mostra as combinações necessárias:

Função	Seleção de modo	Seleção A	Seleção B	Seleção B̄
Soma	1	1	1	0
Subtração	1	1	0	1
Aumento A (estabelecida a primeira entrada de transporte em 1)	1	1	0	0
Aumento B	1	0	1	0

Podemos colocar o sinal de seleção de modo em zero para todas as operações lógicas (elas não exigem a entrada de "transporte da coluna anterior"). Assim, a função XOU pode ser obtida na saída da soma colocando-se a seleção A e a B em HI (alto) e o sinal de seleção de modo em LO (baixo). A função E não pode ser tomada diretamente do circuito existente e exige entradas A e B separadas, com o sinal de seleção E.

Finalmente, a função OU pode ser criada combinando-se as saídas de XOU e de E por meio de uma porta OU. A tabela de validação a seguir demonstra isso:

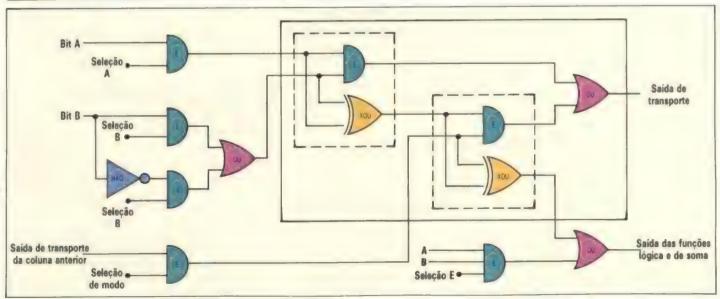
A	B	Saída	Comentários
0	0	0	
0	1	1	Função XOU
1	0	1	
1	1	1	Função E

A próxima tabela mostra como as funções lógicas podem ser produzidas com o uso de diferentes combinações do sinal de controle:

Função	Seleção de modo	Seleção A	Seleção B	Seleção B̄	Seleção E
XOU	0	1	1	0	0
E	0	0	0	0	1
OU	0	1	1	0	1

O diagrama final, abaixo, mostra um estágio de 1 bit da ALU, ilustrando o circuito somador completo e todos os adicionais para os sinais de controle. O circuito completo incorporaria oito circuitos em paralelo. A saída de transporte da oitava coluna é usada como o flag de transporte para o registro de estado do processador.

Este artigo conclui nossa série sobre lógica. Iniciamos o curso com conceitos lógicos abstratos e passamos ao exame de circuitos lógicos simples, bem como dos resultados que fornecem. Em seguida, investigamos circuitos mais complexos, em nível muito próximo ao usado nos computadores.

Finalmente, mostramos como é possível combinar vários circuitos de forma que o microprocessador execute as operações aritméticas e lógicas necessárias. Cada operação é realizada pela colocação dos padrões corretos de 1 e 0 nas linhas de comando da ALU. Esses padrões são na verdade instruções em código de máquina em sua verdadeira forma binária. Assim, estudamos os procedimentos lógicos do hardware no interior de computadores até o ponto em que o software assume o fator de controle.

TRANSPORTE DE VALORES

Dando continuidade a nosso estudo da adição, examinaremos agora o papel que desempenha nessa operação aritmética o flag de transporte e o registrador de estado do processador.

No ASSEMBLY do Z80 e do 6502, a instrução para somar é ADC (Add With Carry, "adicione com transporte"), um mnemônico de grande importância para a programação nessa linguagem. O conceito de bit de "transporte" possui uma significação específica. Consideremos a adição no acumulador de dois números em hex:

```
   A7   =     10100111
 + 3E   =   + 00111110
 ----------------------
   E5   =     11100101
```

Como o acumulador é um registrador de 8 bits, tanto os números a serem adicionados como a própria soma devem estar na faixa que vai de $00 a $FF (caso do exemplo acima) — do contrário, não caberão no acumulador. Isso significaria, então, que estamos restritos a fazer adições em que a soma é menor que $100? Examinemos outra adição no acumulador, a qual viola tal restrição:

```
   FF   =     11111111
 + FF   =   + 11111111
 ----------------------
  1FE   =    111111110
```

Esta é a adição dos maiores números possíveis de um só byte e parece ser uma operação ilegal. Requer um acumulador de 9 bits. A solução para esse dilema está sugerida na formulação do problema — precisamos de apenas um bit a mais no acumulador, para conter o maior resultado da adição de números de um só byte. Esse bit a mais é necessário apenas no total, não nos operandos da adição, e requisitado somente quando há um "transporte" do bit mais significativo do acumulador.

A função do PSR

Esse bit extra é, por isso, conhecido como bit de transporte, e está localizado no registrador de 8 bits associado com o acumulador chamado PSR (Processor Status Register, "registrador de estado do processador"). Esse importante componente conecta-se ao acumulador e à ALU (Arithmetic and Logic Unit, "unidade lógica e aritmética"), de modo tal que os bits individuais do PSR são estabelecidos ou zerados em resposta a qualquer operação do acumulador e dependendo dos resultados da operação. O conteúdo do PSR pode ser encarado como um número simples, mas, em geral, é mais informativo considerá-lo uma matriz de oito elementos, constituída de flags binários; o estado individual de cada um mostra os efeitos particulares da última operação. (Chama-se flag qualquer variável cujo valor não é absoluto, mas sim indicativo do estado, ou valor-verdade, de alguma condição. Uma variável flag costuma ter apenas dois estados ou condições: para cima ou para baixo, ligado ou desligado, 0 ou 1.)

Qualquer operação efetuada no acumulador que ocasione um transporte do oitavo bit coloca o flag de transporte do PSR em 1; uma operação que não cause transporte provoca um reset no flag de transporte, isto é, o recolocará em 0. Isso só se aplica às operações que legitimamente provoquem um transporte. Algumas operações, como carregar no acumulador ou armazenar sinais provenientes dele, não afetam o flag de transporte. Daqui para a frente, neste curso, sempre que aprendermos uma nova instrução da linguagem ASSEMBLY, vamos verificar quais bits do PSR (ou registrador de flags) essa instrução afetará. Estudaremos também os outros flags do PSR, mas vamos primeiro compreender bem a função do flag de transporte.

Ao adicionar dois números de um só byte, não sabemos de antemão quais são; portanto, temos de estar preparados para que a soma exceda $FF; em geral, para isso, reservamos 2 bytes de RAM para conter o resultado de uma adição. Considere, novamente, os dois exemplos prévios de adição:

Números hex		Flag de transporte	Números binários
A7	=		10100111
+ 3E	=		+ 00111110
00E5	=	0 (Não há transporte)	11100101
FF	=		11111111
+ FF	=		+ 11111111
01FE	=	1 (Transporte)	111111110

O resultado da adição é representado em ambos os exemplos como um número de 2 bytes. No primeiro caso, o flag de transporte é recolocado em 0 porque não há transporte proveniente do oitavo bit da soma (o resultado, que ocupa 2 bytes, é $00E5, onde o byte maior é $00). No segundo caso, entretanto, há um transporte do oitavo bit, de modo que o flag de transporte é colocado em 1, e o byte maior do resultado é $01.

Para termos certeza de obter um resultado correto numa adição, devemos, portanto, armazenar o conteúdo do acumulador no menor dos 2 bytes do endereço e o flag de transporte no byte maior. Não há uma instrução em separado para armazenar o flag de transporte, mas o código de

operação ADC foi formulado com vistas a esse tipo de soma: ADC significa, efetivamente, "adicione o operando da instrução ao conteúdo atual do flag de transporte e, então, some esse resultado ao conteúdo do acumulador". A adição é, assim, um processo em dois estágios: o primeiro utiliza o estado atual do flag de transporte e o segundo o atualiza.

Significa então que, antes de iniciar uma adição, devemos levar em conta o estado corrente do flag de transporte, uma vez que ele será somado ao resultado da adição: daí as duas instruções não explicadas nos artigos anteriores, CLC e AND A. A primeira é uma instrução do 6502, e significa "zerar o flag de transporte", fazendo exatamente isso. Na versão Z80, AND A significa "execute a soma lógica (AND) do acumulador consigo mesmo". Embora não tenha sido projetada apenas para zerar o flag de transporte, essa instrução tem tal efeito e mais nenhum outro; por isso, é com freqüência usada como equivalente, no Z80, da CLC do 6502.

Tendo zerado o flag de transporte antes de começar uma adição, devemos em seguida armazenar seu conteúdo. Para tanto, adicionamos o valor imediato $00 ao byte maior do resultado. Isso não afetará o byte se o flag de transporte estiver em 0, mas lhe adicionará 1 se o flag de transporte estiver em 1.

Tudo o que dissemos neste artigo constitui o primeiro método para efetuar adições de um só byte:

1) Zerar o flag de transporte;
2) Carregar o acumulador com um número;
3) Adicionar a este o segundo número;
4) Armazenar o conteúdo do acumulador no menor dos 2 bytes de um endereço;
5) Carregar o acumulador com o conteúdo do byte maior;
6) Somar o valor imediato $00; e
7) Armazenar o conteúdo do acumulador no byte maior.

Eis a conversão desse procedimento para linguagem ASSEMBLY:

COMUM A AMBOS OS PROCESSADORES		
Rótulo	**Diretivo**	**Operando**
BYTE1	EQU	SFF
BYTE2	EQU	SFF
LOBYTE	EQU	SA000
HIBYTE	EQU	SA001
	ORG	SA020

Z80		6502	
Código op	**Operando**	**Código op**	**Operando**
LD	A S00	LDA	#S00
LD	(HIBYTE).A	STA	HIBYTE
AND	A	CLC	
LD	A.BYTE1	LDA	#BYTE 1
ADC	A.BYTE2	ADC	#BYTE2
LD	(LOBYTE).A	STA	LOBYTE
LD	A.(HIBYTE)	LDA	HIBYTE
ADC	A S00	ADC	#S00
LD	(HIBYTE).A	STA	HIBYTE
RET		RTS	

Lembre-se de que os valores dados para BYTE MENOR, BYTE MAIOR e ORG são apenas exemplos — devem-se escolher valores adequados ao seu equipamento. Repare que as duas primeiras instruções do programa carregam $00 em BYTE MAIOR, para que ele não seja alterado por dados aleatórios. Não temos de zerar o BYTE MENOR da mesma maneira porque seu conteúdo inicial será sobregravado pelo byte menor do resultado.

Vale a pena notar as diferenças de abordagem entre as versões Z80 e 6502, como se vê no exemplo. O código do 6502 é fácil de ler, desde que se esteja habituado — os códigos mnemônicos e o uso de "# " indicando dados imediatos tornam claro o significado de cada instrução. A versão Z80 é menos direta porque utiliza o mnemônico LD (load) em todas as transferências de dados, tanto para o acumulador como deste para fora. Tampouco há um símbolo "#" para assinalar os dados imediatos: estes são indicados apenas pela ausência de parênteses no operando. Assim, LD A, BYTE1 significa "carregue o acumulador com o dado imediato BYTE1"; enquanto LD A (HIBYTE) significa "carregue o acumulador com o conteúdo do endereço HIBYTE". Na listagem completa da linguagem ASSEMBLY não há ambigüidade no significado de tais instruções, uma vez que o valor hexadecimal do código de operação identifica exclusivamente a instrução. Pode parecer, entretanto, que isso contorna a questão — o código pode ser único, mas se há uma escolha entre vários códigos únicos, como pode o interpretador ou o programador escolher entre eles? A resposta está no modo de endereçamento, assunto do próximo artigo.

Por fim, devemos observar que o registrador de estado do processador contém outros flags além do de transporte; vamos examiná-los rapidamente agora, retornando a eles mais tarde.

PSR Z80	**S**	**Z**		**H**		**P/V**	**N**	**C**	
Número do bit	7	6	5	4	3	2	1	0	
MSB									LSB
PRS 6502	**S**	**V**		**B**	**D**	**I**	**Z**	**C**	

PSR Bit	Z80
7	(S) = SINAL
6	(Z) = ZERO
5	Desocupado
4	(H) = MEIO-TRANSPORTE
3	Desocupado
2	(P/V) PARIDADE/ESTOURO
1	(N) = SUBTRAÇÃO
0	(C) = TRANSPORTE

6502	PSR Bit
(S) = SINAL	7
(V) = ESTOURO	6
Desocupado	5
(B) = BREAK	4
(D) = MODO BCD	3
(I) = INTERRUPÇÃO	2
(Z) = ZERO	1
(C) = TRANSPORTE	0

Os flags importantes são os de transporte, de sinal e o zero. Vimos que, após uma adição, o flag de transporte contém o valor de transporte do oitavo bit do acumulador. O flag de sinal é sempre uma cópia do oitavo bit (bit 7) do acumulador, sendo o flag zero colocado em 1 se o conteúdo do acumulador for zero e recolocado em 0 se o conteúdo for diferente de zero.

Respostas do exercício anterior

1) Os programas em ASSEMBLY estão no boxe à direita. Repare que os símbolos BYTE1 e BYTE2 são utilizados como dados imediatos e como endereços simbólicos. Como endereços, entretanto, devem ser montados sob a forma de 2 bytes.

2) Está faltando a instrução "retorno da sub-rotina" no fim de ambos os programas. Na versão 6502, o programa completo deveria incluir esta linha:

```
A00D          60          RTS
```

E a versão Z80 precisa desta linha:

```
A00D          C9          RET
```

3) O valor $45 é, em primeiro lugar, carregado no registrador do acumulador como dado imediato; adiciona-se então a ele o número $45, de modo que o acumulador passa a conter o valor $8A. Esse total acumulado é, então, armazenado no endereço $0045 da RAM. Em seguida, adiciona-se o valor $38 como dado imediato no acumulador, de modo que este passa a conter o valor $C2 ($45 + $45 + $38). Esse total é finalmente armazenado no endereço $0038 da RAM.

4) "Dado imediato" é aquele efetivamente armazenado, e não o endereço de um dado. Nas instruções dos exercícios (tais como LD A #$9C e LD A,$E4), os valores $9C e $E4 são os dados a carregar no acumulador. Armazenam-se nas instruções como operandos e compreendem o conteúdo do byte que segue imediatamente o código de operação. Se os dados não estiverem disponíveis, deverão ser então armazenados em outra parte da memória e chamados pelos seus endereços, e não pelos seus valores reais.

5) O valor de BYTE1 é dado como $45, que, convenientemente escrito, fornece o endereço $0045, localizado na página zero da memória.

Decimal para binário

Muitas vezes devemos examinar o conteúdo do PSR e exibir este número como um byte binário em vez de hexadecimal. Segue-se uma sub-rotina em BASIC para conversão de decimal-binário. Você poderá utilizá-la em conjunto com seus programas em ASSEMBLY ou em BASIC.

```
7000 REM *** DECIMAL PARA BINARIO *****
7001 REM * CONVERTE UM NUMERO N (<256) *
7002 REM * NUM NUMERO BINARIO DE       *
7003 REM * 8 CARACTERES EM B$          *
7010 B$ = ""
7020 FOR D = 8 TO 1 STEP - 1
7030 LET N1 = INT (N/2)
7040 LET R = N - 2 . N1
7050 LET B$ = STR$(R) + B$
7060 LET N = N1
7070 NEXT D
7080 RETURN
```

Variações

No Commodore 64, troque a linha 7050 da sub-rotina por:

```
7050 B$ = MID$(STR$(R),2) + B$
```

Endereço da localização	Código de máquina	Linguagem ASSEMBLY		
6502				
0000		START	EQU	SA000
0000		BYTE1	EQU	S45
0000		BYTE2	EQU	S38
0000			ORG	START
A000	A9 45		LDA	#BYTE1
A002	18		CLC	
A003	69 45		ADC	#BYTE1
A005	8D 45 00		STA	BYTE1
A008	69 38		ADC	#BYTE2
A00A	8D 38 00		STA	BYTE2
Z80				
0000		START	EQU	SA000
0000		BYTE1	EQU	S45
0000		BYTE2	EQU	S38
0000			ORG	START
A000	3E 45		LD	A,BYTE1
A002	A7		AND	A
A003	CE 45		ADC	A,BYTE1
A005	32 45 00		LD	(BYTE1),A
A008	CE 38		ADC	A,BYTE2
A00A	32 38 00		LD	(BYTE2),A

LDA — CARREGAR O ACUMULADOR — 6502

Imediato —A9 (2 bytes)

Carrega-se no acumulador o conteúdo do byte que sucede o código de operação.

EFEITO SOBRE O PSR

S V B D I Z C

MSB [X][ ][ ][ ][ ][ ][X][ ] LSB

Exemplo:

LOCALIZAÇÃO	CÓDIGO DE MÁQUINA	LINGUAGEM ASSEMBLY
A000	A9 3F	LDA #S3F

	ANTES	DEPOIS
PSR	????????	0?????0?
A	??	3F

Memória de dados

Memória do programa: A9 SA000; 3F SA001

LD A, — CARREGAR O ACUMULADOR — Z80

Imediato —3E (2 bytes)

Carrega-se no acumulador o conteúdo do byte que sucede o código de operação.

EFEITO SOBRE O PSR

S Z H V N C

MSB [ ][ ][ ][ ][ ][ ][ ][ ] LSB

NÃO TEM EFEITO

Exemplo:

LOCALIZAÇÃO	CÓDIGO DE MÁQUINA	LINGUAGEM ASSEMBLY
A000	3E 9B	LD A,S9B

	ANTES	DEPOIS
PSR	????????	????????
A	??	9B

Memória de dados

Memória do programa: 3E SA000; 9B SA001

LDA — CARREGAR O ACUMULADOR — 6502

Da memória —AD (3 bytes)

Carrega-se no acumulador o conteúdo da localização da memória cujo endereço sucede o código de operação.

EFEITO SOBRE O PSR

S V B D I Z C

MSB [X][ ][ ][ ][ ][ ][X][ ] LSB

Exemplo:

LOCALIZAÇÃO	CÓDIGO DE MÁQUINA	LINGUAGEM ASSEMBLY
6F00	AD 07 B3	LDA SB307

	ANTES	DEPOIS
PSR	????????	1?????0?
A	??	D2

Memória de dados: SB307 D2

Memória do programa: AD S6F00; 07 S6F01; B3 S6F02

LD A, — CARREGAR O ACUMULADOR — Z80

Da memória —3A (3 bytes)

Carrega-se no acumulador o conteúdo do byte que sucede o código de operação.

EFEITO SOBRE O PSR

S Z H V N C

MSB [ ][ ][ ][ ][ ][ ][ ][ ] LSB

NÃO TEM EFEITO

Exemplo:

LOCALIZAÇÃO	CÓDIGO DE MÁQUINA	LINGUAGEM ASSEMBLY
6F00	3A E9 F4	LD A,(SF4E9)

	ANTES	DEPOIS
PSR	????????	????????
A	??	8C

Memória de dados: SF4E9 8C

Memória do programa: 3A S6F00; E9 S6F01; F4 S6F02

SISCO

Sempre buscando soluções que atendam às necessidades de seus clientes, a Sisco alcançou alto nível de tecnologia e know-how no desenvolvimento e implantação de sistemas.

Fundada em 1978, a Sisco - Sistemas e Computadores reuniu um núcleo de especialistas capaz de desenvolver toda uma linha de produtos de hardware e software e assegurar completo suporte técnico, treinamento e manutenção para seus sistemas.

Somando atualmente 715 funcionários, a empresa produz, com tecnologia própria, além de micros e supermicrocomputadores, o sistema modular MIC 1000, que pode funcionar como central independente, interface remota ou local; o terminal de vídeo semigráfico TV 3000, com recursos de gerenciamento na tela; o processador de comunicação síncrona PCS; e o conversor assíncrono/síncrono para ligação de dois computadores AS-01.

Uma das características da Sisco reside no intenso controle de qualidade realizado a cada etapa da produção. Esse fator é responsável pelo número crescente de empresas de grande porte entre seus clientes.

Supermicro
Desenvolvido pela Sisco, o MB 8000 S/M é um supermicrocomputador com capacidade de multiprogramação, que opera com até oito terminais de vídeo. O MIC 1000 (embaixo) é um sistema modular para supervisão e controle de processos exclusivo da Sisco.

Suporte de software

A preocupação da Sisco não é apenas lançar no mercado produtos de boa qualidade, mas orientar o usuário durante as fases de instalação, implantação e manutenção dos sistemas. Segundo Carlos Masotti, um de seus diretores, "A venda dos produtos só começa após a instalação do sistema".

No período de pré-instalação, a empresa faz um completo levantamento das necessidades reais do cliente, visando identificar a melhor configuração e os aplicativos mais adequados. Após a implantação do sistema, a Sisco treina o usuário e seus funcionários, desenvolve e converte aplicativos, dá assessoria e consultoria em análise de sistemas, oferecendo suporte técnico de software e de hardware.

Outros recursos

A Sisco já desenvolveu mais de duzentos programas para quase todas as áreas profissionais. Comércio, bancos e instituições financeiras, geração e transmissão de energia, são algumas das áreas abrangidas.

A empresa conta com ampla equipe de assistência técnica, com centros de atendimento espalhados por diversos Estados brasileiros e pelo interior paulista.

Toda essa estrutura é apenas o começo de uma empresa que promete muito ao mercado da informática. Além dos micros, a Sisco conta ainda com o lançamento de um minicomputador, o Sistema 4000, destinado a aplicações de multiprogramação e multiusuário, possuindo ampla capacidade de armazenamento de dados e saída para até nove terminais de vídeo. Esse míni segue a filosofia da boa interação com o usuário.

COMANDO MENTAL

Jogos do pensamento
O acoplamento mental elimina o estágio mecânico intermediário da conversão dos impulsos do jogador em sinais significativos para o computador. Colocando usuário e máquina em relação direta, o acoplamento mental possibilita programas mais interativos e entradas mais rápidas.

O controle de um computador pelo pensamento deixou de ser ficção científica. Conheça o fenômeno que fundamenta essa tecnologia e o dispositivo que acopla o sistema nervoso ao computador.

Desde que surgiram os primeiros computadores, o teclado tem sido o principal dispositivo de entrada de informações. Talvez seja o método mais eficaz de se introduzir textos e, em menor grau, números — pelo menos até que se desenvolvam sistemas confiáveis de reconhecimento de voz e destinados a objetivos gerais. Entretanto, por mais adequado que seja para se fornecerem textos à máquina, o teclado não é, necessariamente, o melhor dispositivo para se introduzirem outros tipos de informação. Se precisamos inserir dados direcionais com rapidez, como em vários jogos, damos preferência ao joystick ou ao paddle; no caso de dados posicionais, como numa seleção de opções num menu, o mouse, a caneta óptica ou a tela sensível ao toque mostram-se em geral mais apropriados; e a leitora de código de barras é o mais conveniente para longos números ou códigos seriais.

Mas todos esses dispositivos apresentam um inconveniente: são indiretos. Nos jogos, o joystick é mais adequado que o teclado (para fazer a nave subir, por exemplo, empurramos a empunhadura para a frente), mas ele ainda representa um hiato entre o que pretendemos e o que de fato acontece. Por exemplo, pensamos "para cima", traduzimos isso mentalmente como "empurrar para a frente" e só então empurramos. Para uma interação mais rápida e direta entre nós e o computador, seria preciso eliminar essa mediação: por que não apenas pensar "para cima" e obter do aparelho pronta resposta ao nosso pensamento?

Assim, jogaríamos Invasores do Espaço apenas ordenando mentalmente "Para cima", "Para baixo", "Vire", "Fogo", e assim por diante; ou, então, escreveríamos uma carta apenas pensando nas palavras. Claro que o processador de textos telepático ainda não existe, mas jogar Invasores do Espaço por controle mental já é uma realidade, graças à noção de "acoplamento mental". Segundo esta, empregamos o pensamento — mais precisamente, as alterações fisiológicas resultantes das mudanças nos padrões de pensa-

Notas mentais

A foto ilustra uma demonstração de um protótipo do sistema RGP funcionando com um computador Apple. Sistemas semelhantes já foram desenvolvidos para uso no IBM PC e no Commodore, mas é bastante provável que o software efetivo ainda leve mais tempo. Roger Dilts, presidente da Behavioral Engineering e autor de software para RGP, tem especial interesse na possibilidade de combinar o acoplamento mental com a NLP (Neuro-Linguistic Programming, "programação neurolingüística"), um ramo da psicologia dedicado ao estudo do aprendizado. Com a utilização da técnica de RGP para monitorar o estado mental do usuário, o software pode medir o grau de tensão do mesmo e ajustar o andamento do programa. Roger Dilts acredita que os programas educacionais — para citar um exemplo — serão capazes de avaliar as reações emocionais dos estudantes ao material apresentado. Se este for demasiado complexo, o computador captará a tensão emocional provocada, permitindo alterar o ritmo do programa ou simplificar o tema.

mento — para controlar um dispositivo eletrônico, que funciona como uma interface para um computador, tal como o joystick ou outro dispositivo de entrada.

O acoplamento mental baseia-se no conhecido fenômeno RGP (Reação Galvânica da Pele), pelo qual mudanças no estado emocional se manifestam na condutividade elétrica da pele. Eletrodos são colocados na pele do usuário e ligados a um medidor de resistência. Introduzem-se os sinais do medidor no micro através do conector de expansões, e um software especial os interpreta e processa. A empresa Behavioral Engineering (Engenharia Comportamental), da Califórnia, baseou tanto jogos como softwares aplicativos nessa técnica, inclusive uma versão simplificada do Invasores do Espaço. Importantes empresas de computação vêm trabalhando há anos nesse setor, a Atari, por exemplo, mas só recentemente alcançaram algum êxito.

No jogo convencional, controlamos a nave na órbita de um planeta. O objetivo é atingir naves inimigas sem que a nossa o seja, e sem que ela se espatife contra o planeta. Na versão produzida pela Behavioral Engineering, controlamos apenas a altitude da nave, mas com uma diferença: pelo pensamento! A empresa projetou uma interface de RGP para o Apple II em que o usuário tem apenas de colocar os dedos indicador e médio num aparelho parecido com um mouse. Este, medindo a resistência elétrica entre os dois dedos, envia o resultado para o computador. O software foi planejado para que um aumento na tensão comande a nave para cima,

Acoplamento mental

Esse princípio baseia-se num fenômeno conhecido por três nomes alternativos: RGP (Reação Galvânica da Pele); RPG (Reflexo Psicogalvânico); e RED (Reflexo Eletrodérmico). Preferimos adotar o primeiro termo. A RGP refere-se a mudanças na condutividade elétrica da pele que correspondem a alterações no estado emocional de uma pessoa. Experimentos mostram que, quanto mais tensa uma pessoa, mais baixa se torna a resistência elétrica de sua pele. A aplicação mais conhecida da RGP está nos polígrafos, os chamados detectores de mentiras.

Embora se observe o fenômeno da RGP em animais e seres humanos desde o século XIX, pouco se conhece sobre sua causa. Segundo a teoria inicial, o suor produzido pela excitação ou ansiedade age como condutor eletrolítico, baixando assim a resistência da pele. Experiências mais recentes, porém, lançaram dúvidas sobre essa teoria tão simplista. Seja qual for a razão do efeito, sabe-se que a RGP relaciona-se diretamente ao grau de tensão no sistema nervoso simpático. Este depende do sistema nervoso central e, portanto, do cérebro. Daí ser possível o controle de um computador pelo pensamento. Alterações na atividade cerebral acarretam mudanças no estado do sistema nervoso central; isso modifica o estado do sistema nervoso simpático, resultando em variação na resistência elétrica da pele. E a alteração numa corrente elétrica é a base do funcionamento de qualquer dispositivo de entrada.

e uma redução a leve para baixo. O objetivo é controlar esses movimentos e alinhar a nave com os alvos que se aproximam.

O termo "acoplamento mental" talvez seja enganoso, pois essa ligação se dá não pela mente, mas pelo sistema nervoso. Eis a questão: se ambos são interdependentes, faz sentido separá-los? Muitas pessoas mostram-se capazes, após cerca de 20 minutos, de exercer um razoável grau de controle, quase sempre sem saber como o conseguem. Umas, de forma consciente, tensionam e relaxam levemente o corpo; outras simplesmente pensam "Para cima" ou "Para baixo" e deixam o sistema nervoso fazer o resto.

O acoplamento mental tem outras aplicações mais práticas. Uma pessoa totalmente paralítica ainda conta com o recurso de gerar, de modo consciente, efeitos de RGP, embora não controle os músculos. Um dispositivo de RGP ligado a um paralítico foi acoplado com sucesso a um robô Topo, através de um computador Apple, dando a essa pessoa a possibilidade de controlar o robô. Uma outra aplicação prática estaria no espaço sideral, sob gravidade zero. É extremamente difícil operar controles mecânicos sem a gravidade para contrabalançar a energia aplicada. Um dispositivo de RGP poderia ser o substituto perfeito para muitos controles mecânicos.

Um software que capta o ânimo

Entre as aplicações exóticas está a captação da disposição de ânimo. Os softwares educacionais, em particular, poderiam se beneficiar com a retroalimentação do estado interior do usuário por meio da RGP. O indivíduo usaria em torno do pulso uma faixa com eletrodos, que o software calibraria no estado de relaxamento normal. Se, num dado momento, o aparelho registrasse acentuado aumento de tensão, significando estar o usuário em dificuldades, ele ajustaria o andamento do programa. Softwares comerciais seriam capazes de detectar a tensão mental de usuários — e de sugerir um descanso.

A rapidez do desenvolvimento dessa tecnologia depende não só da criação de dispositivos mais precisos e de respostas mais rápidas, como também de nossa habilidade em interpretar os efeitos. O primeiro problema está em que a RGP ocorre 2 segundos após o evento e leva de 2 a 10 para desaparecer. Os dispositivos atuais superam em parte o problema, medindo o ritmo de mudança da reação e não sua intensidade. O segundo problema está na imprecisão das nossas deduções a partir dos dados fornecidos pela RGP. Uma queda brusca na resistência elétrica da pele indica tensão ou estresse, mas não se pode precisar se a causa é agradável ou dolorosa.

Como se vê, apesar de todos os problemas, a RGP descortina possibilidades entusiasmantes. E a Atari não é a única a levá-la a sério: a Commodore ofereceu a respeitável soma de 2 milhões de dólares à Behavioral Engineering pelos direitos do dispositivo e do software de RGP — devidamente recusados!

O detector de mentiras

Numa investigação ou julgamento, um dos maiores problemas enfrentados pela polícia e sistema judicial é saber se um suspeito ou uma testemunha está mentindo. Embora um perito possa detectar pistas mínimas (como alterações na coloração da pele e na respiração, por exemplo), a medição desses fatores não está definida nem catalogada — e muito menos é admissível como prova. Por isso se pesquisa no sentido de desenvolver meios objetivos de distinguir entre declarações falsas e verdadeiras. Um dos resultados nesse sentido foi o polígrafo, ou detector de mentiras.

O polígrafo, na verdade, consiste apenas num medidor de resistência elétrica. Supõe-se que o indivíduo sofrerá aumentos significativos de tensão ao dizer uma mentira ou ouvir palavras associadas ao crime, e que isso se revelará por uma queda na resistência elétrica da pele.

O polígrafo vem sendo objeto de muita controvérsia. É defendido pela polícia, nos EUA em especial, mas seus críticos alegam que o nosso conhecimento sobre o fenômeno RGP é insuficiente para uma interpretação segura e coerente. Além do fato de indivíduos diferentes reagirem de forma também diferente, sejam eles culpados ou inocentes.

Na conquista de um emprego
Às vezes se usa a RGP na avaliação de candidatos a empregos, alimentando um detector de mentiras na análise das respostas do entrevistado. Na foto acima, o interessado recebe os resultados do teste que acabou de fazer.

Mentiras inofensivas
O detector de mentiras mede mudanças de RGP causadas pelas reações emocionais do indivíduo. Mas é pouco provável que esse aparelho forneça respostas tão precisas quanto as que vemos abaixo. Reações emocionais variam conforme a sensibilidade do indivíduo a certos temas (se os acha constrangedores, por exemplo) e não dependem apenas da falsidade ou veracidade das respostas.

CENTOPÉIA FAMINTA

Diversos programas criados para o TK 85 são adaptáveis ao TK 90X. Analisamos aqui duas versões de um mesmo jogo para esses micros. Aproveite para fazer suas próprias modificações.

Nosso programa é um jogo de ação onde você dirige uma centopéia esfomeada, em busca de comida, dentro de um jardim. O controle se faz com as teclas [5], [6], [7] e [8] (as setas de controle do cursor), e você deve evitar que a centopéia "morda" a si mesma ou os tijolos que delimitam o jardim. Se isso acontecer, ela morrerá.

Os alimentos são pequenas larvas (números) que surgem ao acaso na superfície do jardim. Podem ser mais ou menos gordas: quanto mais gorda, maior o número por ela representado, e cada número que a centopéia devora a faz crescer proporcionalmente em segmentos.

Mas o jardim permite que a centopéia tenha no máximo quatrocentos segmentos. Como ela já "nasce" com um, significa que poderá crescer até 399. Seu objetivo final, então, consiste em fazer com que todo o jardim seja ocupado pela centopéia.

A versão para o TK 85

As linhas 20, 30 e 40 desenham na tela os "tijolos". As variáveis L e C armazenam a posição do último segmento da centopéia, e as variáveis Y e X armazenam a posição da cabeça. O string A$ armazena os movimentos que a cauda deve realizar para chegar à posição da cabeça. As outras variáveis servem como indicadores de passagem, atuando como contadores ou como auxiliares.

As linhas de 40 a 130 apenas inicializam algumas variáveis. A 150 mostra na tela um segmento do corpo da centopéia. As linhas de 160 a 190 sorteiam um número e uma posição no vídeo, e a linha 200 fixa a posição em que aparecerá o número sorteado. Feito isso, o programa é desviado para a sub-rotina da linha 490, que verifica qual caractere existe nessa posição do vídeo e armazena seu código na variável E. Dependendo desse código, o número sorteado será ou não exibido.

As linhas de 250 a 300 permitem que o controle se faça com as teclas [5], [6], [7] e [8]. A linha 310 fixa a posição em que será exibida a cabeça da centopéia. Depois disso, o programa se desvia para a sub-rotina da linha 490.

As linhas 330 e 340 verificam se a centopéia simplesmente andou, se ela mordeu um tijolo, se devorou a si mesma ou a uma larva. Em cada caso, algumas das linhas entre 350 e 480 são executadas.

As linhas 350, 355 e 360 são executadas se a centopéia morre. Já as 370, 380 e 390 são executadas no caso de ela devorar alguma larva. A linha 400 exibe a cabeça no vídeo e a 420 apaga a cauda. O movimento da cauda é produzido nas linhas 430 e 440.

A variável B indica se vai ou não haver crescimento, Z conta quantos segmentos a centopéia aumentou, e W armazena o total de alimentos na superfície do canteiro.

As linhas 410 e 470 estão relacionadas com o conteúdo de B, e as 460 e 480 desviam incondicionalmente o programa para a linha 150, onde um outro segmento do corpo da centopéia é exibido.

O programa deve ser executado no modo slow, extremamente lento quando o micro tem mais de 2 Kbytes de RAM. Isso ocorre porque, quando o aparelho possui menos de 3,25 Kbytes de RAM disponíveis, gerencia o armazenamento do vídeo de uma forma mais econômica.

Caso seu micro tenha 16 Kbytes, para aumentar-lhe um pouco mais a velocidade, você poderá "enganá-lo" alterando o valor da variável do sistema que armazena o endereço do último byte existente na RAM: a RAMTOP ou RTP. Para isso, basta comandar diretamente (antes de digitar o programa):

```
POKE 16388,200
```

e

```
POKE 16389,11
```

Isso fará o micro "pensar" que está apenas com cerca de 3 Kbytes de RAM, tornando-se um pouco mais rápido. Quanto mais à esquerda estiver a imagem no vídeo, menos memória será necessária para o armazenamento. Devido a isso, o programa para o TK 85 (e similares) constrói a moldura descentralizada, mais à esquerda.

Devido à lentidão, impõe-se que os números sorteados na linha 170 (larvas) sejam sempre superiores a 3, para que o crescimento se acelere mais.

A versão para o TK 90X

O TK 90X possui muitos recursos visuais e sonoros que podem ser usados em quase todos os tipos de programas. Em nosso caso específico, esses recursos são essenciais para transformar o programa Centopéia (lento e um pouco pobre no

TK 85) num programa ao nível de qualquer bom videojogo.

Uma grande vantagem do TK 90X sobre o TK 85 é patente: a velocidade de execução do programa. O TK 85 possui duas velocidades de processamento. No modo slow, o vídeo se apresenta permanentemente ao usuário, o que torna a execução lenta. No modo fast, a execução acontece mais rápida, porém o vídeo não é mostrado.

Num jogo de ação, precisa-se utilizar o modo slow. O TK 90X funciona sempre mostrando o vídeo, porém com uma velocidade de processamento próxima à do fast.

Outro fator que também torna o programa lento no TK 85 é a utilização da sub-rotina da linha 490, necessária para checar a posição do vídeo em que ocorrerá a próxima impressão. No TK 90X existe uma instrução extremamente simples, a SCREEN$, utilizada nas linhas 200 e 310, que substitui com vantagem a sub-rotina usada no TK 85.

O uso das cores também é muito simples. Os comandos BORDER, PAPER e INK são usados na linha 10 de modo a gerar uma tela branca com caracteres pretos e uma moldura vermelha.

Acrescentamos som ao programa com o comando SOUND. A linha 250 gera um sinal cada vez que o teclado é verificado, e a 350 gera um sinal de "morte" da centopéia (sua marcha fúnebre!).

Outro grande melhoramento possibilitado pelo TK 90X é o uso de caracteres determinados pelo usuário. Foram definidos três deles: um para os "tijolos" do jardim (linha 30), um para o corpo da centopéia (linha 150) e outro para sua cabeça (linha 400). Os três estão ilustrados em detalhes na figura para que possam ser reproduzidos.

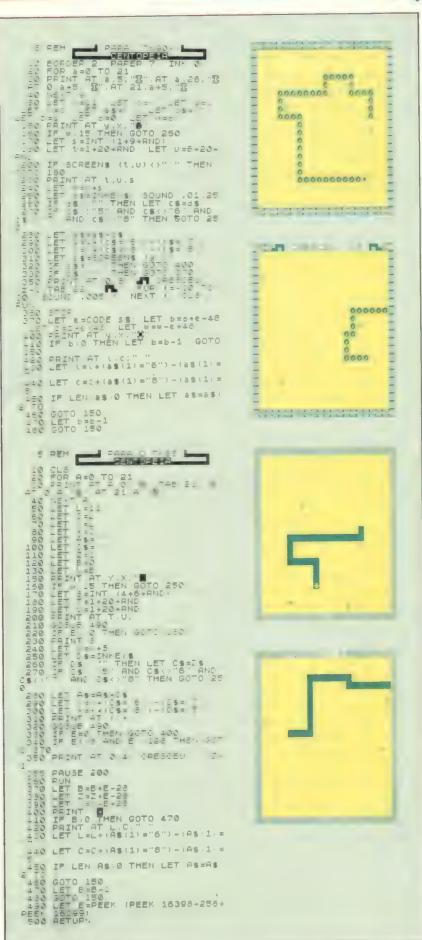
Por fim, como fica evidente, muitas vezes utilizamos linhas com multiinstrução, isto é, várias instruções numa mesma linha do programa.

Como no TK 90X não temos problemas com a velocidade de processamento, as larvas são representadas por qualquer número entre 1 e 9. Além de muito mais rápido, mais estético e mais interessante, o programa no TK 90X também fica mais compacto.

Outras opções

Existem várias maneiras de se produzir um programa como este; utilizamos apenas uma delas. No caso específico do TK 85, certamente não é a melhor (seria muito mais rápido imprimir no vídeo, colocando, por meio do comando POKE, caracteres diretamente na memória de vídeo).

No TK 90X, entretanto, a lógica usada é bastante viável (nesse caso, acabaria sendo bem mais complicado utilizar o POKE para colocar caracteres diretamente no vídeo).

Constituiria um exercício bastante instrutivo tentar tornar o programa para o TK 85 mais rápido usando POKE ao invés de PRINT. Ainda mais instrutivo seria adaptar esse programa para o TK 90X, dada a sua complexidade. Fica como sugestão para um fim de semana chuvoso.

SOLUÇÃO PARA COMPATIBILIDADE

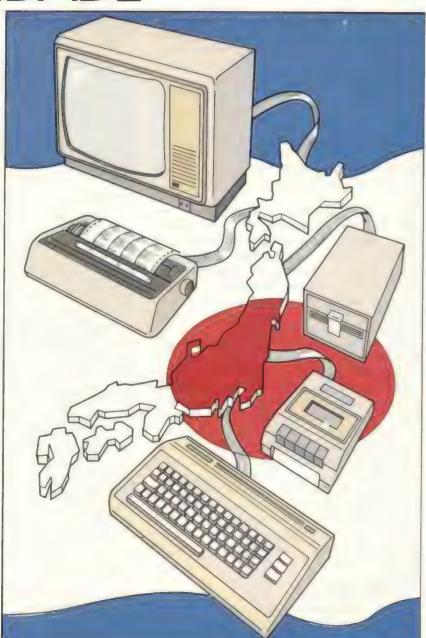
A incompatibilidade entre os diversos equipamentos é o grande problema dos usuários: programas para determinada máquina não rodam em outra. O padrão MSX tenta solucionar isso.

Rumo ao sucesso
O padrão MSX é a rota do Japão para entrar nos mercados mundiais de computação. Se obtiver sucesso, poderá dominar esse mercado, da mesma maneira que dominou as áreas de aparelhos de som e das câmaras fotográficas.

Para entender por que as diversas marcas de microcomputadores têm arquiteturas tão diferentes, precisamos nos reportar à história dessa indústria. Os primeiros microprocessadores a causar grande impacto no mercado, o Intel 8080 e o Motorola 6800, tiveram suas instruções fixadas desde o início; o problema da compatibilidade surgiu em virtude da versatilidade dos processadores. Por exemplo, para enviar um byte de dados para uma saída, qualquer microcomputador usaria a mesma instrução, desde que utilizasse o mesmo processador. Mas a saída poderia ter qualquer endereço, entre centenas ou milhares.

Os primeiros micros foram produzidos por diversos fabricantes "de fundo de quintal"; dessa forma, não houve uniformidade sobre a localização das portas de entrada e saída no mapa da memória. Um fabricante poderia escolher o endereço 255 para a saída de impressora paralela, enquanto outro escolheria o endereço 254 para o mesmo fim. Essa situação tornou-se ainda mais caótica com o advento dos chips controladores de telas gráficas e de efeitos sonoros. Qualquer programa que utilizasse os recursos especiais de um determinado computador certamente não rodaria em outro, a menos que sofresse consideráveis modificações.

Se houvesse, na época, uma indústria mais forte, que impusesse um padrão, a situação seria diferente. Mas isso não aconteceu: no início da microcomputação, havia inúmeros pequenos fabricantes, cada um com seu estilo e seus padrões. Não somente as máquinas acabavam se apresentando física e eletronicamente diferentes, como também as linguagens de programação utilizadas eram incompatíveis entre si. Mesmo o BASIC nunca foi padronizado, pois na década de 70 os profissionais da informática não o levaram muito a sério, visto que o consideravam linguagem para principiantes.

No final da década de 70 e início da de 80, os microcomputadores se desenvolveram a enorme velocidade. Os projetos pioneiros — o Apple, por exemplo — incorporaram refinamentos como gráficos e cores; mas para pôr em prática essas inovações, os fabricantes desenvolveram suas próprias versões do BASIC, que assim proliferaram. Se por um lado essa expansão possibilitou ampla liberdade de escolha ao consumidor, por outro gerou muitas frustrações aos proprietários, aos fabricantes de micros e aos projetistas de software. O orgulhoso proprietário de um SORD ficaria desesperado para obter o último jogo da moda; mas, se o programa fosse escrito para o Sinclair, ele teria de esperar muito pela versão compatível com sua máquina.

O fabricante que tem um novo micro a lançar sabe que existe pouquíssimo software para seu equipamento até que haja um número substancial de usuários. Mas, com pouco ou nenhum

software disponível, o potencial de vendas de sua máquina é muito limitado, pondo em risco todo o investimento feito no desenvolvimento do produto.

Se você cria software comercial, suas vendas limitam-se (na melhor das hipóteses) às pessoas que possuem o modelo de computador para o qual se destina o programa. Suponhamos que você crie um jogo de aventuras para o TK 90X chamado Masmorras do Inferno, com personagens como o Fantasma Invisível, o Carcereiro Sádico e incontáveis Antros do Terror para apanhar os incautos. A equipe de pesquisa de mercado afirma ser imenso o potencial de vendas, desde que se lance o produto a um certo preço e se vendam, no mínimo, 65.000 cópias. Entretanto, o número de proprietários do TK 90X não basta para garantir a vendagem; é necessário pelo menos mais uma versão. O custo de produção de novas versões, para, digamos, o CP 400 e o Apple, aumentaria o valor unitário em 20%; a esse preço, as vendas ficariam abaixo de um nível economicamente viável. É esse o dilema que frustra muitos projetistas de software em potencial.

O problema da compatibilidade de software não foi ignorado pela indústria de computadores. A óptica individualista do Ocidente não favorece a padronização; mas, no Extremo Oriente, os fabricantes preferem que as coisas sejam mais sistemáticas, organizadas e uniformes, especialmente quando isso se transforma em lucro.

A ASCII/Microsoft está tentando organizar o caos. A companhia resulta de uma fusão da Microsoft Corporation com a ASCII, uma bem-sucedida editora japonesa de revistas populares. Como linha secundária, a ASCII também publica software comercial e, quando a Microsoft se impôs o desafio de entrar no "impenetrável" mercado japonês, a ASCII foi a primeira opção de parceria num empreendimento conjunto: a Microsoft possuía a tecnologia e a ASCII a experiência de marketing.

A Microsoft Corporation construiu sua reputação sobre a versão do BASIC mais aceita como padrão, o MBASIC, adotado por fabricantes de micros do mundo todo. Mesmo assim, não havia garantias de que um programa em MBASIC rodasse em todos os computadores que usassem essa linguagem, pois sempre que houvesse recursos especiais haveria incompatibilidade de hardware.

O BASIC da Microsoft foi vendido para numerosos fabricantes japoneses através da ASCII/Microsoft. Mas isso também não resolveu o problema de compatibilidade de hardware e software. Como solução, a ASCII/Microsoft criou, em cooperação com os principais fabricantes japoneses, um padrão de aceitação internacional. O resultado desses esforços foi o chamado padrão MSX, que inclui requisitos básicos de hardware (centrados no microprocessador Z80 e alguns outros chips) e também uma linguagem padronizada. No final de 1985, os micros MSX já faziam bastante sucesso no mercado internacional; no Brasil, os pioneiros foram o Gradiente Expert e o Sharp Hot-Bit.

Especificações do MSX

O BASIC MSX assemelha-se muito ao MBASIC (da Microsoft), mas com várias melhorias que aproveitam as modernas possibilidades gráficas e sonoras. Seguem-se os novos comandos: SCREEN, para especificar o modo de tela, o tamanho dos sprites, o bip das teclas, a velocidade de transmissão de cassete e as opções da impressora; LOCATE, que posiciona os caracteres na tela; COLOUR seleciona uma das dezesseis cores de linha e de fundo; PUT SPRITE estabelece atributos dos sprites; CIRCLE faz círculos e elipses; DRAW executa desenhos; LINE traça retas entre coordenadas especificadas; e PAINT preenche figuras com uma determinada cor. Há também o comando KEY, para alocar strings a teclas de função. Outros comandos permitem introduzir valores na RAM de vídeo (VPOKE), inscrever valores nos registradores do chip de efeitos sonoros (SOUND) e controlar o motor do gravador cassete (MOTOR).

O MSX, entretanto, envolve não apenas a padronização do software, mas também especificações quanto ao hardware. A CPU é um processador Z80 funcionando a 3,58 MHz. Deve haver no mínimo 32 Kbytes de ROM para armazenar o software MSX, e no mínimo 8 Kbytes de RAM. Não há limite máximo para a capacidade de ROM e de RAM. Um computador MSX deve incorporar um chip controlador de vídeo TMS9918A, da Texas Instruments (ou equivalente), e um AY38910—um chip gerador de som com três vozes. O vídeo deve apresentar 24 linhas, com 32 ou 40 colunas. No momento, não há recursos para apresentação em 80 colunas. Quanto à resolução, exige-se 256 x 192 pixels.

Escolheu-se a fita cassete como meio básico de armazenamento de programas e dados; ela deve usar o sistema de codificação FSK, com velocidade de transmissão de 1.200 ou 2.400 bits por segundo. O teclado também é padronizado, e inclui teclas de função e caracteres gráficos. No Brasil, a Gradiente optou por dotar o Expert com o gerador internacional de caracteres, que permite grafar todas as letras e acentos da língua portuguesa da mesma forma que nas máquinas de escrever.

Uma entrada para cartuchos de ROM permite a utilização de softwares aplicativos. Há um conector de cinqüenta pinos para entrada e saída, e interface para dois joysticks.

Os formatos dos discos também são padronizados, assim como o sistema operacional em disco, o MSX-DOS, que equivale funcionalmente ao MS-DOS e permite a leitura de arquivos de dados gravados com esse sistema. Parece ser compatível, ainda, com o popular sistema operacional CP/M 2.2. Também se estabeleceram formatos para os disquetes de 3 1/2 (Sony e Gradiente), 5 1/4 e 8 polegadas.

Isso tudo significa que programas escritos para uma máquina MSX e armazenados em qualquer disco certamente rodarão sem problemas em qualquer outro computador da linha MSX, com plena utilização de seus recursos gráficos e sonoros. As vantagens são óbvias, tanto para o fabricante como para o consumidor.

O padrão MSX corre apenas alguns riscos. O primeiro é que qualquer "padrão", uma vez adotado, não conseguirá incorporar as inovações da área. Se, por exemplo, surgir no mercado um novo chip controlador de vídeo com recursos muito superiores, os programas MSX não poderão aproveitá-lo e deixarão o campo aberto aos concorrentes capazes de utilizá-lo.

O segundo risco está nos microprocessadores de 8 bits, dos quais o mais popular é o Z80, pois por princípio são incapazes de endereçar diretamente mais de 64 Kbytes de memória principal; também não podem processar de uma só vez dados de valor superior a 256.

À época do lançamento do MSX, parecia provável que nos próximos dez anos o mercado de micros passasse a ser dominado pelos microprocessadores de 16 bits. Se o padrão MSX conseguisse o sucesso esperado, sua conseqüência mais positiva seria a de alertar os fabricantes que a padronização deve vir cedo — e não tarde — no projeto de qualquer inovação.

O MSX poderia facilitar a venda imediata dos produtos com base no Z80, mas dificilmente isso seria vantajoso a longo prazo. Seu real impacto estava em convencer o resto do mundo de que a padronização é importante.

No campo dos micros de 16 bits, a IBM provou que "contra a força não há argumentos", ao impor, com seu computador pessoal, na prática, um padrão. O MSX seria capaz de fazer o mesmo para os micros de 8 bits? Até fins de 1985, ele foi apoiado por muitos fabricantes, como a Yamaha, JVC, Hitachi, Sony, Sanyo, National, Pioneer, Canon, Fujitsu e Mitsubishi do Japão, a americana Spectravideo, a coreana Daewoo e Gradiente e Sharp no Brasil. Contudo, nenhum fabricante europeu entrara para o clube. Somente o tempo e a reação do mercado diriam, nos próximos anos, se havia necessidade de mais padronização, ou do tipo de inovação individualista que se espera de empresários como os pioneiros da computação, que, com seu engenho, criaram em precárias condições os primeiros microcomputadores.

Convenções do projeto
Para que os programas e dispositivos adicionais sejam compatíveis com todos os sistemas MSX, o hardware de um micro MSX segue regras estritas. Uma vez que a máquina se conforma à configuração básica mostrada aqui, os projetistas podem acrescentar dispositivos extras.

PADRÃO MSX

O MSX é o padrão para micros adotado por mais de doze fabricantes japoneses, alguns bastante conhecidos pelos seus produtos eletrônicos. Vamos examinar dois desses equipamentos: o Hit-Bit e o HX-10.

O padrão MSX prescreve: o microprocessador a ser usado (Z80), a quantidade mínima de ROM (32 Kbytes) e de RAM (8 Kbytes), o tipo de chip de gráficos e som, as teclas que devem estar presentes (embora o layout do teclado possa variar), o número mínimo de interfaces (e seus modelos), as telas gráficas e de texto, e a linguagem BASIC contida na ROM.

Sendo o MSX um projeto padronizado, é de se esperar que todos os micros MSX se assemelhem. Os diversos modelos, no entanto, admitem flexibilidade: a quantidade de memória que pode ir além daquele mínimo, o tipo de teclado e o número extra de interfaces. Assim, a maioria dos equipamentos MSX atingiu especificações acima dos requisitos mínimos.

Entre os modelos estrangeiros, o Hit-Bit da Sony e o HX-10 da Toshiba possuem um teclado de boa qualidade, mas há quem ache as teclas demasiadamente sensíveis ao toque. Os dois micros vêm com 64 Kbytes de memória principal e 16 Kbytes adicionais de RAM, exclusivos para o funcionamento da tela. Esse total de 80 Kbytes é superior ao fornecido pela maioria dos micros. Ambos os modelos possuem interfaces para impressora com saída paralela padrão Centronics e duas saídas para joysticks — itens freqüentemente opcionais.

Esperava-se que os micros MSX se tornassem máquinas baratas, mas as flutuações no câmbio e o aumento dos custos de fabricação empurraram os preços para cima. Outra razão para o encarecimento foi a pressa com que se procurou colocar essas máquinas no mercado europeu.

Uma das primeiras coisas que nos chama a atenção quando ligamos um micro MSX é uma linha com palavras na parte inferior da tela. São comandos da linguagem BASIC, tais como RUN, CLOAD, LIST etc. Cinco teclas de função apresentam as palavras mais usadas, que, na tela, servem de rótulo para as teclas de função, de modo que o usuário não precisa memorizar a função de cada tecla.

Essas teclas são automaticamente definidas quando se liga a máquina, mas é fácil mudar a definição usando-se o comando KEY. Embora só existam cinco teclas, podem ser acessadas até dez funções. Para tanto, pressionam-se, ao mesmo tempo, a função desejada e a tecla [Shift]. Esta, quando acionada, muda as palavras na parte inferior da tela, apresentando as novas funções designadas às teclas. Cada atribuição de função pode conter até quinze caracteres, mas apenas os sete primeiros aparecem na tela.

Sistema padronizado
O Toshiba HX-10 possui uma interface para impressora com saída paralela padrão Centronics e saídas para joystick (duas) e para cartucho de ROM. O BASIC do MSX lida com os joysticks da mesma maneira como comanda os movimentos do cursor, de modo que jogos desenvolvidos para um tipo de controle podem automaticamente utilizar outro. *À esquerda*, o Sony Hit-Bit e seus periféricos.

O teclado e o editor da tela funcionam concatenados para facilitar a edição. Quatro teclas movimentam o cursor pela tela, podendo-se fazer

alterações em qualquer lugar da mesma, bastando escrever em cima dos caracteres existentes. A inserção e supressão de caracteres requerem um único toque. No HX-10, as teclas do cursor têm o mesmo tamanho das outras; já no Hit-Bit, esse conjunto é bem maior e bastante diferenciado. Como tais teclas são intensamente usadas, esse tipo de layout revela-se muito conveniente.

Assim como o hardware e o software do MSX estão cheios de recursos extras, seu BASIC inclui comandos como AUTO e RENUM — que facilitam a codificação — e contém vários comandos para manipulação de som, gráficos e interrupções. Três deles são básicos para a geração de gráficos. O comando LINE traça uma linha ligando dois pontos, mas pode também ser usado para desenhar um quadrado — basta acrescentar a letra B (box) logo após as coordenadas. Adicionando-se as letras BF (box fill), obtém-se um quadrado preenchido com uma cor. O comando CIRCLE é usado para desenhar elipses e arcos, assim como círculos básicos. E PAINT preencherá totalmente qualquer contorno, mesmo os mais complicados, com uma cor.

O BASIC do MSX inclui muitas outras características úteis, mas o conjunto mais sofisticado de comandos, para "manipulação de interrupções", pode não ser apreciado de imediato. São comandos muito úteis para programar gráficos em alta velocidade. Há várias situações em que um programa deve realizar uma tarefa, ao mesmo tempo que verifica se outra coisa está acontecendo.

Exemplo típico disso são jogos do tipo Invasores do Espaço. O programa deve manter os alienígenas em constante movimento pela tela e, simultaneamente, verificar se o botão de "fogo" foi acionado. Precisa, portanto, fazer as duas

Desfraldando a bandeira

Padrão MSX

CPU	Z80A, de 3,58 MHz
RAM	8 Kbytes, no mínimo
ROM	32 Kbytes, incluindo o BASIC
TELA	Dezesseis cores, gráficos de 256 x 132 pixels, 32 sprites, texto de 24 linhas x 40 colunas (ou 24 x 32); chip de vídeo TI 9918 ou equivalente
SOM	Três canais, podendo ser acessado diretamente pelos comandos do BASIC; chip controlador de som AY 3-8910
INTERFACES	Para cartucho MSX, monitor de tevê, impressora e para cassete
TECLADO	Tipo QWERTY, com teclas especiais de função, quatro de movimentação do cursor e dez de funções programáveis

Variações MSX

SONY HIT-BIT	Software de banco de dados gravado em ROM, saída RGB, expansões opcionais de 4 Kbytes de RAM
TOSHIBA HX-10	Bus de expansão, duas saídas para joystick
YAMAHA CX-5	Miniteclado musical e software para a interface MIDI
PIONEER	Interface controladora de videodisco a laser
SANYO MPC100	Caneta óptica opcional e software
JVC HC7GB	Saída RGB
SVI 728 SPECTRAVIDEO	Teclado numérico completo

Apesar de o padrão MSX requerer um mínimo de 8 Kbytes de memória, todos os fabricantes acima fornecem 64 Kbytes de RAM para o usuário, mais 16 Kbytes para a RAM de tela.

coisas ao mesmo tempo, trocando rápido de tarefa.

A solução do MSX é designar certas coisas como "eventos". São fornecidas instruções ao computador para que esteja atento a um evento. Se algum ocorre, o computador chama automaticamente uma sub-rotina para ocupar-se dele.

O modo gráfico do MSX pode mostrar dezesseis cores com uma resolução de 256 x 192 pixels. Definem-se até 32 sprites de 8 x 8 pixels (ou 16 sprites de 16 x 16 pixels; ou 8 sprites de 32 x 32 pixels). Para o aproveitamento máximo dos sprites, o BASIC do MSX inclui um conjunto completo de comandos dedicados a eles; por exemplo, SPRITE para definir um sprite, e PUT SPRITE para posicioná-lo em qualquer lugar da tela.

Já se encontra disponível um grande número de software em cartuchos. E a promessa de compatibilidade parece estar sendo cumprida — os aplicativos para o HX-10 rodam perfeitamente no Hit-Bit e vice-versa. Isso vale tanto para aplicativos em cartucho como para programas em cassete. Depois de anos de sistemas não compatíveis, parece quase mágico poder tirar um cartucho de um micro e usá-lo em outro. Os fabricantes do padrão MSX, confiantes no sucesso do equipamento graças a essa característica, procuram colocar bem rápido no mercado uma ampla gama de aplicativos.

No Brasil, foi com o padrão MSX que a Gradiente e a Sharp ingressaram, em 1985, na área da informática. Seus micros, o Expert e o Hot-Bit, respectivamente, visam sobretudo o mercado educacional. É a contribuição brasileira a esse padrão de computadores bem equipados, divertidos de usar e de preço bem razoável. E que vêm cumprindo à altura as promessas dos fabricantes.

MSX TOSHIBA HX-10

MICROPROCESSADOR

Z80A.

CLOCK

3,58 MHz.

MEMÓRIA

64 Kbytes de RAM (28 Kbytes disponíveis para o BASIC), 16 Kbytes de RAM de tela, 32 Kbytes de ROM, incluindo o BASIC.

VÍDEO

Modo texto: 4 linhas x 40 colunas; modo gráfico: 256 x 192 pixels, dezesseis cores e até 32 sprites.

TECLADO

Estilo máquina de escrever, com 68 teclas, sendo quatro para o cursor e cinco de funções programáveis.

LINGUAGEM

BASIC expandido da Microsoft.

INTERFACES

Saídas para impressora paralela (padrão Centronics), monitor, tevê, áudio, cassete, joystick, duas aberturas para cartucho de ROM, bus de expansão.

DOCUMENTAÇÃO

Guia para instalação e guia de referência de programação em BASIC. Ambos são bem feitos, mas não suficientemente completos.

FORÇA BRUTA

O favorito
O Cray-1 *(foto acima)* oferece uma enorme capacidade de processamento, sob o controle de um computador frontal como um IBM, DEC, ou similar. A arquitetura do Cray-1 *(diagrama abaixo)* apresenta 32 Mbytes de memória principal e 4.888 bytes de espaço de registradores. Todo esse sistema aloja-se em nada menos que 3.400 painéis de circuitos impressos, com mais de 90 km de fios para conectá-los.

Muitas áreas da ciência exigem processamento rápido de imensas quantidades de dados. Para esse fim, são usados "supercomputadores", como o Cray-1, que estudamos neste artigo.

A potência de um computador, em termos simples, é função de seu tamanho de palavra (quantos bits são processados por vez), velocidade de transferência de dados, tamanho da memória principal e velocidade de ciclos da unidade central de processamento. Nos micros, as funções principais da CPU residem num único chip microprocessador. Este pode ser o conhecido Z80, os Intel 8088 e 8086, e o Motorola M68000. Todos utilizam a tecnologia MOS (Metal Oxide Semiconductor, "semicondutor de óxido metálico") para os circuitos lógicos e a memória do chip. O processamento e transferência de dados se dá em paralelo, a 8 ou 16 bits por vez. As freqüências de clock dos microprocessadores variam de 1 a 12 MHz.

Todos esses recursos, mesmo parecendo avançados, não são suficientes para que os micros dêem conta do colossal volume de processamento de dados necessário para aplicações como animação de imagens, dinâmica dos fluidos e previsão de tempo.

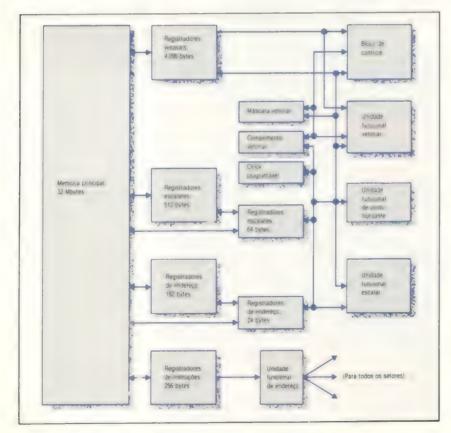
Vejamos um exemplo concreto. Imagine os requisitos necessários para fazer um filme com gráficos animados gerados por computador, com resolução de 6.000 x 6.000 pixels e 24 fotogramas, ou quadros, por segundo. Como os gráficos são animados, os pontos na imagem deslocam-se de quadro em quadro, de modo que a posição de cada ponto terá de ser calculada a cada novo fotograma. Isso significa 864 milhões de cálculos por segundo, cada um bastante complexo e envolvendo dezenas ou centenas de instruções em código de máquina. Isso resulta em bilhões de instruções por segundo.

Os supercomputadores, nome dado às máquinas de maior porte, funcionam de modo muito semelhante aos micros. As instruções e os dados vêm da memória; o processador manipula os dados de acordo com as instruções e armazena os resultados na memória. A diferença principal está na escala e na velocidade de execução dessas operações, bem como na arquitetura dos componentes do hardware.

Vejamos um exemplo típico de supercomputador: o Cray-1 S/4400, fabricado pela Cray Research. O primeiro Cray-1 foi instalado em 1976 e vem se firmando como o mais famoso e o mais popular supercomputador de grande porte. A CPU do Cray-1 ocupa um gabinete semicircular de quase 2 m de altura, parecido com um sofá curvo (o sistema de resfriamento e a fonte de alimentação ficam sob os "assentos"). Sua velocidade resulta da lógica de semicondutores bipolares, que são transistores "comuns", ao contrário do MOS, CMOS, NMOS, FET e tantos outros tipos. A lógica bipolar e a memória provêm de mais de 200.000 circuitos integrados, dispostos em 3.400 painéis; usam-se mais de 90 km de fios para interconectá-los.

O tamanho de palavra usado nas computações do Cray-1 é de 64 bits, ou seja, oito vezes mais que os 8 bits processados por um Z80. O clock do sistema tem uma velocidade de 80 MHz (80 milhões de ciclos por segundo); como resultado, uma adição em 64 bits leva apenas 37,5 nanossegundos, ao passo que uma adição em 8 bits, num Z80 operando a 4 MHz, leva 1,75 microssegundo.

A memória principal no Cray é parte integrante da CPU. Ela contém 32 Mbytes, dispostos sob a forma de 4.194.304 palavras, e transfere até 2,56 bilhões de bytes por segundo. Como se pode ver, o Cray possui um poderoso conjunto de registradores na CPU. Há 72 registradores de en-

dereço (cada um com 24 bits de comprimento), 72 registradores escalares (cada um com 64 bits de comprimento) e oito registradores vetoriais (cada qual com 64 palavras de comprimento). O espaço total dedicado aos registradores na CPU do Cray é, portanto, de 4.888 bytes; em comparação, o microprocessador Z80 possui apenas 26.

O terminal frontal

Ao contrário da maioria dos computadores, que constituem unidades completas e independentes, os computadores Cray foram projetados para ser extensões de um computador de grande porte ou um minicomputador já existente. Este, chamado computador frontal, funciona como uma interface, aceitando entradas de terminais ou de leitoras de cartão e enviando as saídas para periféricos como impressoras ou fitas magnéticas.

Entre a CPU do Cray e o computador frontal fica o subsistema de entrada e saída do Cray, projetado para agilizar o fluxo de dados para a CPU. Há também uma interface, que adapta o sistema Cray às características específicas do IBM, DEC, Data General ou outro computador usado como terminal frontal. O subsistema de entrada e saída do Cray consiste em dois a quatro processadores de entrada e saída, sendo cada um deles, sozinho, um potente minicomputador.

Veja no boxe o diagrama de blocos da CPU do Cray-1, para uma idéia mais precisa de sua sofisticação. Poderíamos ainda citar o conjunto de instruções, as treze unidades funcionais que operam em paralelo e o processamento vetorial, que permite trabalhar com até 64 pares de operandos com uma única instrução. Em resumo, não exageramos ao dizer que o Cray é extremamente poderoso.

Aplicações dos supercomputadores

Meteorologia
Previsões de tempo resultam da coleta de dados em âmbito mundial, comunicações via satélite e modelagem computacional. Um "modelo" elaborado por computador requer centenas de milhões de cálculos para se obterem previsões em questão de horas. E apenas computadores extremamente potentes como o Cray processam tamanha quantidade de dados com a rapidez necessária.

Dinâmica dos fluidos
Ciência fundamental no projeto de automóveis econômicos, sistemas de refrigeração de usinas nucleares, engenharia aeronáutica e muitas outras aplicações. Prever o movimento de fluidos exige a análise de imensas quantidades de dados, e os problemas se avolumam quando se necessita de resultados em tempo real. Cada partícula num fluido — e há incontáveis trilhões delas — atua sobre todas as outras do conjunto em movimento. Assim, o cálculo do comportamento de um sistema de fluidos exige uma capacidade maciça de processamento de dados.

Previsões econômicas
É difícil construir modelos econômicos, pois, como na dinâmica dos fluidos, pequenas mudanças num elemento repercutem sobre todo o sistema. Mesmo modelos simples são de enorme complexidade. Também aqui se recorre a computadores extremamente rápidos e potentes, para que os analistas não tenham de esperar meses pelos resultados.

Alta velocidade

Para comparar a capacidade de processamento de um supercomputador Cray-1 à de um micro baseado no Z80, considere a produção de uma seqüência de 10 minutos de filme com gráficos animados em alta resolução. Vamos admitir uma resolução de 6.000x6.000 pixels, e 24 fotogramas por segundo; cada pixel precisa ser calculado para cada novo fotograma. Na versão para o Z80, necessita-se de cem instruções em código de máquina, com um tempo de execução médio de dezenove ciclos de clock (4,75 microssegundos). Para o Cray-1, com suas poderosas instruções de processamento vetorial, diremos que são necessárias 25 instruções em código de máquina (um cálculo pessimista), com um tempo médio de execução de quatro ciclos de clock (50 nanossegundos). Para a produção desse filme, um micro baseado no Z80 levaria $6.000^2 \times 24 \times 60 \times 10 \times 100 \times 4{,}75 \times 10^{-6} = 2{,}4624 \times 10^8$ segundos, ou seja, 7,8 anos. Um Cray-1 levaria $6.000^2 \times 24 \times 60 \times 10 \times 25 \times 50 \times 10^{-9} = 6{,}48 \times 10^5$ segundos, ou seja, apenas 7,5 dias.

Revolução visual
Os gráficos gerados por computador estão revolucionando a produção de vídeos e filmes. Criam-se imagens, como as mostradas, por meio da moderna tecnologia de micros e minicomputadores. Trabalhando em maior escala, usou-se o Cray-1 para gerar mais de 20 minutos de imagens no filme *The last starfighter (O último combatente do espaço)*, executando cálculos que ocupariam um micro de 8 bits por mais de quinze anos.

O PASSO DO ROBÔ

Caminhando

Andar é um movimento da maior complexidade. Para fazer isso, um robô deveria ter tantas articulações quanto o homem e estar programado para dar seus passos exatamente assim: deve mudar seu centro de gravidade, inclinando o corpo para a frente e levantando a perna que ficou atrás. O ombro gira e os braços balançam, equilibrando o corpo sobre o pé de apoio, com a ajuda dos artelhos, até que o outro pé vá à frente dar apoio.

Para ser útil, um robô precisa de movimentos. Mas conseguir isso envolve a resolução de vários problemas, como a construção de pernas, pés, braços, mãos, articulações e "músculos".

Movimentar o corpo, para um ser humano, é mais do que trivial: é natural. Mas reproduzir numa máquina esses movimentos aparentemente tão simples constitui um desafio que nem sempre pode ser vencido: alguns dos movimentos humanos — como o andar — apresentam tal complexidade que colocam grandes obstáculos à sua reprodução mecânica.

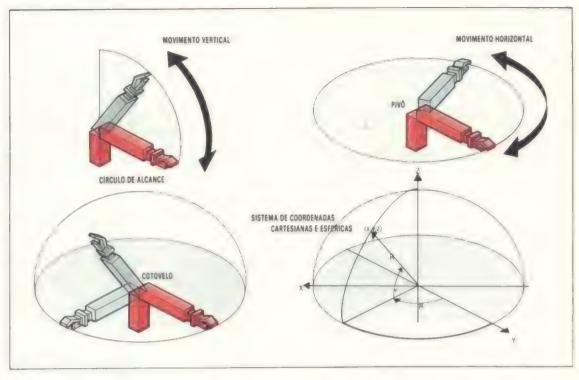
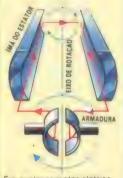
E como fazer, então, para dotar os robôs de mobilidade, isto é, capacidade de mover-se de um lugar para outro? Bem, devemos nos lembrar, primeiro, de que a maior parte dos trabalhos que se pretende de um robô exige que ele fique parado. Assim, fazê-lo andar acaba não sendo tão importante.

Mas há tarefas que exigem mobilidade. Por exemplo, robôs que lidam com explosivos para as forças armadas de diversos países têm de movimentar-se. Como?

Há diversas maneiras de fazer um robô andar, especialmente se ele tiver pernas. Nesse caso, estas podem se mover para a frente e para trás, simulando o andar humano, arrastando os pés pelo chão. Esses pés podem ter rodas com catracas, de modo que só se movam para diante. E aí o problema está resolvido — só que fica difícil manobrar o robô.

A solução seria fazê-lo erguer os pés do chão para caminhar. Mas aí surge outro problema: ele terá de equilibrar-se em uma das pernas enquanto anda. As soluções são complicadas, mas não impossíveis. Um robô assim até subiria escadas — só ficaria difícil montar o mecanismo para ele descobrir quando a escada acaba.

Muito mais simples é usar esteiras, como os tanques de guerra — com a desvantagem de que os robôs não são tão grandes, perdendo, portanto, para os tanques em estabilidade. Além disso, os movimentos carecem de precisão. Para manobrar, uma das esteiras tem de ficar parada, enquanto a outra continua a mover-se. O ideal seria que a esteira parada apenas girasse sobre o solo, em torno de seu centro. Mas isso não acontece, e ela acaba se deslocando, o que, mesmo sendo pouco, significa um problema. No caso do tanque, o deslocamento é corrigido pelo homem que o dirige.

Rotações possíveis
Mesmo o mais simples robô deve poder movimentar o braço como nestas ilustrações. Para isso, o cotovelo é articulado, permitindo um movimento vertical semicircular (de 90°) e horizontal circular (360°). Assim, o braço alcança qualquer ponto num hemisfério cujo raio tem seu tamanho. E a localização desse ponto é dada por coordenadas do seguinte modo (veja o desenho): o ângulo H corresponde ao do arco tangente x/y; o ângulo V é o do arco seno z/R (arc sen z/R). Assim, o programa que leva o braço a um determinado ponto deve conter as coordenadas x,y,z desse ponto. Esses valores são transformados em movimentos dos servomotores, que dirigem então o braço até o ponto exato de trabalho.

Passo a passo

Em qualquer motor elétrico uma corrente na bobina do rotor cria um fluxo magnético contrário ao do estator. É justamente essa oposição de forças que produz movimento no rotor.

O rotor de um motor a passo pode ter centenas de bobinas. Passando-se a corrente de alimentação de uma para outra obtêm-se movimentos pequenos de grande precisão.

Por essas e outras razões, robôs usam rodas, que produzem movimento suave e são mais fáceis de controlar. Ficam faltando, então, um meio de o robô saber exatamente onde está e um mecanismo para movimentar as rodas com precisão.

Para o primeiro problema, a solução está no plano cartesiano, definido por dois eixos de coordenadas. Isso basta para localizar o robô e determinar os pontos para os quais ele tem de se deslocar.

Fazer isso com precisão, contudo, exige dispositivos especiais. Não podemos, nesse caso, usar motores elétricos comuns para movimentar as rodas, as quais, cortada a alimentação, ainda giram pelo menos meia volta, e isso constitui um grave inconveniente para aplicações que exigem precisão. Nesse caso, a solução está no uso de motores a passo, ou graduais, que produzem movimentos precisos, de qualquer amplitude. É esse o tipo de motor usado nas tartarugas, que têm rodas e um suporte para caneta, podendo, assim, executar desenhos.

Mas tudo se complica bem mais quando se trata de reproduzir os movimentos da mão e do braço do ser humano, necessários para que o robô trabalhe. Em primeiro lugar, deve-se estabelecer um sistema para determinar a posição do braço em qualquer instante; depois, esse braço precisa de uma estrutura e de "músculos".

A primeira questão tem solução também pelos eixos de coordenadas; mas nesse caso há um complicador, que é um terceiro eixo (já que o movimento do braço se faz em três dimensões). Com isso, pode-se descrever a posição do braço em qualquer ponto do espaço em que ele atua. É possível, então, contruir um robô cujo braço se desloque ao longo desses três eixos de coordenadas — para a frente e para trás, para cima e para baixo, para a direita e para a esquerda. Fica meio desajeitado, mas funciona bem para trabalhos numa área limitada.

Esse sistema funciona bem para robôs fixos a um suporte, mas não tem a mesma eficiência em outras situações. Para estas, há métodos de descrever o espaço e os pontos dentro dele. Um desses usa como modelo um cilindro e suas coordenadas. Imagine uma lata cilíndrica. Localiza-se qualquer ponto dentro da lata, associando: a distância a que está do centro do círculo que forma o fundo da lata (essa distância é um segmento de raio); o ângulo que ele forma em relação a determinado ponto de referência no círculo; e a altura em relação ao fundo da lata. Com essas três informações, há uma localização exata.

Um sistema parecido tem uma esfera como modelo de espaço. A diferença com o anterior está em que, ao invés de usar a altura em relação ao fundo, a informação necessária consiste no ângulo formado entre o ponto, o centro da esfera e seu diâmetro horizontal.

O processo mais comum para o posicionamento exato do braço de um robô, no entanto, usa somente ângulos: um para descrever a rotação na base, outro para a elevação do braço e um terceiro para uma segunda articulação.

Força muscular

O sistema de coordenadas adotado acabará determinando o tipo de estrutura do braço, faltando, então, os "músculos" que vão produzir o movimento. Podem ser hidráulicos, elétricos ou pneumáticos.

Se a opção for pelos motores elétricos, estes deverão ser de passo, ligados direta ou indiretamente às partes que formam o braço. Um siste-

ma melhor seria imitar os animais, cujos músculos contraem-se e distendem-se. Isso pode ser conseguido por meio de pistões acionados por líquido (hidráulicos) ou ar comprimido (pneumáticos), atuando sobre cada articulação. Como os sistemas hidráulicos suportam grandes pressões e o volume do fluido não se altera (o que acontece com o ar), eles são mais indicados para ambientes industriais, onde os robôs precisam de mais força.

Esses músculos dão ao robô movimento linear; para obter movimento circular usa-se uma espécie de turbina sobre a qual o fluido exerce pressão — o resultado assemelha-se ao de um motor de passo.

Resolvido o problema do braço, fica faltando o da "mão" — para que o braço, depois de colocado na posição pretendida, execute um trabalho. É necessário, primeiro que ela tenha os movimentos fundamentais da mão humana: para cima e para baixo, para os lados e um mínimo de rotação. Um robô capaz de fazer esses movimentos tem seis graus de liberdade na mão. Pode-se também, construí-los com menos graus, mas manter a mobilidade implica dar mais liberdade às outras articulações do braço.

A mão do robô

Tudo isso definido, fica faltando apenas estabelecer como será a mão do robô. O ideal, naturalmente, é fazê-la tão parecida com a do homem quanto possível (e até existem robôs assim). O formato mais comum consiste num "polegar" e dois "dedos", habilitando o robô a apanhar objetos como o homem.

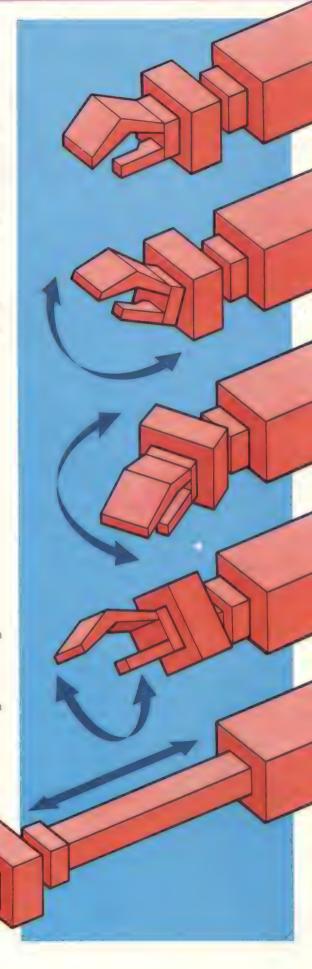
A energia para movimentar essa mão pode ser qualquer das três já citadas — elétrica, pneumática ou hidráulica —, dependendo da tarefa que o robô vai executar. Se ele tiver de trabalhar com objetos grandes e pesados, provavelmente precisará de musculatura hidráulica. Para outras aplicações, porém, motores elétricos ou a ar comprimido bastarão — especialmente se essas aplicações não exigirem precisão absoluta.

Haverá casos em que o robô não precisará de uma mão ou qualquer coisa parecida. Um robô de solda, por exemplo, só requer o equipamento de soldagem ligado diretamente ao punho. Existem robôs capazes até mesmo de escolher e trocar uma ferramenta do punho, de acordo com o tipo de trabalho a executar. Podem, digamos, descartar uma chave de fenda e inserir uma pistola de pintura — desde que o encaixe seja simples —, o que os torna extremamente versáteis.

A grande maioria, contudo, costuma trabalhar o tempo inteiro apenas com uma ferramenta — como os robôs de solda usados na indústria automobilística, que têm características totalmente distintas dos usados em pinturas. Seria inútil um robô de solda capaz de substituir ferramentas, já que seu trabalho é contínuo e exercido sempre no mesmo lugar. Mesmo porque, na fase de soldagem o carro ainda não está pronto para receber a pintura.

O pulso
Dotar um robô de pulso articulado melhora muito seu desempenho, mas complica bastante as transformações exigidas no sistema de coordenadas. E isso ainda é pouco se comparado aos problemas de peso, resistência, precisão e flexibilidade do pulso. Para o software, o maior problema está em decidir qual dos possíveis modos de pegar o objeto será o mais simples.

Flexibilidade
A capacidade de movimentação do pulso humano é limitada exatamente pelos mesmos fatores que restringem os movimentos do pulso de um robô: um meio termo entre resistência e tamanho das articulações com a necessidade de peso reduzido e flexibilidade. Essas articulações não aumentam a área de acesso, mas melhoram a capacidade do braço na manipulação de objetos.

Extensão e giro
Esses movimentos são necessários aos robôs de aplicação industrial, pois aumentam a área de acesso e a capacidade de manipular objetos, com a vantagem de serem mais fáceis de implementar do que os ilustrados acima (com movimentos para os lados, para cima e para baixo).

MUDANÇA DE ENDEREÇO

A linguagem ASSEMBLY deve grande parte de sua versatilidade às diferentes maneiras de calcular endereços na memória. Examinamos aqui os endereçamentos direto, indireto e indexado, bem como suas combinações.

Qualquer instrução em linguagem ASSEMBLY relaciona-se implícita ou explicitamente a um conteúdo da memória. Uma vez que um byte só se diferencia de outro por seu endereço, toda instrução em ASSEMBLY refere-se, portanto, a um ou mais endereços. Às vezes a referência é direta e óbvia, como em LDA $E349, que significa "carregar o acumulador com o conteúdo do endereço $E349". Essa instrução menciona sem ambigüidade o endereço $E349 e sua relação com o acumulador (cujo endereço é um nome, e não um número).

Outras instruções, contudo, referem-se a um endereço de maneira menos clara; por exemplo, RET, que significa "retornar de uma sub-rotina". À primeira vista, aqui não há menção a nenhum endereço; mas RET, na realidade, significa "o endereço da próxima instrução a ser executada é o lugar de onde se chamou a última subrotina". Aqui não se menciona o endereço cujo conteúdo será alterado (ou seja, o contador de programas — o registrador que contém o endereço da próxima instrução a ser executada). Tampouco há referência ao endereço do novo conteúdo. Essas duas instruções representam modos de endereçamento totalmente diferentes.

Até agora, vimos instruções com dois diferentes modos de endereçamento: o imediato (como em LD A,$45 ou ADC #$31) e o direto absoluto (como em STA $58A7 ou LD ($696C),A). Esses modos, "naturais", parecem abranger todos os casos possíveis, exceto os implícitos, como RST ou RET; há, porém, outras possibilidades, que examinaremos agora.

Endereçamento da página zero

Usamos esse endereçamento, ou endereçamento curto, sempre que uma instrução se refere a um endereço entre $0000 e $00FF. Todos os endereços nessa faixa têm $00 como byte mais alto e, portanto, ficam na página zero da memória. Tais instruções necessitam de apenas 2 bytes — um para o código de operação e outro para o byte mais baixo do endereço. Quando a CPU detecta um endereço de um só byte num ponto onde costuma haver dois, assume que o byte mais alto é $00 e, assim, refere-se diretamente à página zero. A vantagem desse processo está na velocidade de execução: o acesso aos dados da página zero é bem mais rápido que a qualquer outra página, exatamente porque requer um endereço de apenas 1 byte.

Os microprocessadores Z80 e 6502 diferem muito quanto ao uso do modo de página zero. A linguagem ASSEMBLY do 6502 usa esse modo em qualquer instrução que elabore os endereços à RAM (tal como LDA), e todos os 256 bytes da página zero ficam disponíveis para a CPU. No ASSEMBLY do Z80, por outro lado, somente uma instrução — RST, "recomeçar" (restart) ou "inicializar" (reset) — utiliza o modo de página zero, e calcula somente para oito determinadas posições dessa página. Como a instrução RST é bem específica em seu endereçamento, requer um código de operação de apenas 1 byte, que inclui o endereço; isso torna sua execução muito rápida.

Pode parecer absurdo comparar a velocidade das instruções em linguagem ASSEMBLY, quando sabemos que o tempo de execução da mais lenta é medido em microssegundos. Entretanto, às vezes a economia de 1 microssegundo por instrução representa a diferença entre o sucesso e o fracasso de um programa. Por exemplo, os jogos que mostram gráficos animados em alta resolução, coloridos e tridimensionais, envolvem milhões de instruções por operação em tela, e devem executar esses comandos o mais rápido possível para obter movimentos suaves e contínuos. Cortar 1 microssegundo de uma operação é muito importante quando ela faz parte de um loop a ser executado 64.000 vezes seguidas.

Endereçamento indexado

O modo indexado é vital para a programação em ASSEMBLY, pois permite a elaboração de estruturas matriciais de dados. Sem tais estruturas, os programas se restringem ao endereçamento de posições de memória uma a uma; foi o que fizemos até agora em nossos programas. Por meio da indexação, entretanto, manipulam-se muito mais dados, e o programador conta com quantidade bem maior de recursos.

Os elementos essenciais do modo indexado são o endereço-base e o *índice*. Vamos supor que queremos montar uma tabela de dados — os códigos dos caracteres ASCII, por exemplo — em bytes consecutivos. O endereço-base da tabela é o do primeiro byte. A partir disso, podemos nos referir a qualquer byte subseqüente da tabela por sua posição em relação ao endereço-base; o primeiro byte fica na posição zero, o segundo na

Endereçamento indexado

VERSÃO EM LINGUAGEM ASSEMBLY

1. BADDR = endereço-base de A$( )
2. Carregar com zero o registrador indicador
3. Começar o loop
4. Carregar o acumulador com BADDR modificado pelo registrador indicador
5. Somar $20 ao acumulador
6. Armazenar o acumulador em BADDR modificado pelo registrador indicador
7. Registrador indicador = registrador indicador + 1
8. Registrador indicador = $0A? — Não: voltar a "Carregar o acumulador..."; Sim: Parar

VERSÃO EM BASIC

```
FOR K = 0 TO 9
C$ = A$(K)
C$ = CHR$(ASC(C$) + 32)
A$(K) = C$
NEXT K
STOP
```

Suponhamos uma matriz A$ de strings em BASIC, cujos dez elementos são caracteres maiúsculos que desejamos converter para minúsculos. Na maioria das máquinas, essa matriz será armazenada como uma tabela de 10 bytes consecutivos no espaço de armazenamento de strings. Escrevemos um programa em código de máquina para fazer a conversão, utilizando o endereçamento indexado; o programa em BASIC equivalente ilustra essa técnica.

Aqui, o nome da matriz, A$( ), indica ao interpretador BASIC o endereço inicial de seus elementos, enquanto BADDR faz o mesmo em ASSEMBLY. Também o contador de loops, K, indica um elemento de A$( ) por vez, da mesma forma que em ASSEMBLY o registrador indicador modifica BADDR.

posição um, o terceiro na posição dois, e assim por diante. A posição de um byte com relação ao endereço-base da tabela chama-se índice, e calcula-se o endereço absoluto de qualquer byte da tabela pela soma do endereço-base com o índice do byte. Se construirmos um loop num programa em ASSEMBLY, e usarmos o contador do loop como índice, poderemos endereçar cada byte da tabela seqüencialmente, da mesma forma como acessamos os elementos de uma matriz em BASIC usando um loop FOR-NEXT.

Uma vez mais, as versões do ASSEMBLY do Z80 e do 6502 diferem quanto ao endereçamento indexado. O chip do 6502 contém dois registradores de um só byte, chamados X e Y; cada um dispõe de um índice que modifica um endereço-base. Isso limita o comprimento da tabela a 256 bytes, que é o maior número representado por um byte. Já o chip do Z80 contém dois registradores de 2 bytes, IX e IY, que contêm o próprio endereço-base e podem ser incrementados ou decrementados para indicar os sucessivos bytes da tabela. Sendo registradores de dois bytes, IX e IY podem conter o endereço de qualquer byte a que a própria CPU tem acesso. Seu conteúdo também se modifica por um índice de um só byte.

Endereçamento indireto

Envolve o uso de endereços indicadores. Para compreender melhor esse conceito, imaginemos um grupo de pessoas que formou um cineclube e se encontra todas as semanas para assistir a um filme. Este pode estar passando em um dos vários cinemas da cidade. Então, escolhido o filme da semana, o presidente escreve a hora e o lugar da projeção num cartaz e afixa-o na vitrine de uma loja no centro da cidade. Os membros do cineclube não sabem qual será o cinema escolhido, mas sabem onde fica a loja, e esta lhes "indica" o local correto. O endereço da loja é, indiretamente, o endereço do cinema.

No modo de endereçamento indireto escrevem-se instruções que contêm o endereço de um indicador e que atuam sobre o conteúdo da po-

Indicador
Exemplo de endereçamento indireto encontrado no cotidiano. Na foto, o quadro indicador dos trens, com destinos e plataformas, contém dados necessários a quem viaja. A seta "Train indicator (indicador de trens) mostra onde encontrar o quadro; assim, ela indiretamente endereça os dados. Em ASSEMBLY o endereçamento indireto significa que o endereço no operando é o do byte que armazena os dados; o do operando, um indicador que aponta para o endereço real

sição para onde o indicador aponta (e não sobre o conteúdo do próprio indicador). As vantagens desse modo são consideráveis, especialmente quando combinado com o tipo indexado. Suponhamos, por exemplo, que você escreva uma rotina em ASSEMBLY que pesquise, numa tabela de dados, um certo caractere, e retorne com sua posição indexada. Suponhamos ainda que você queira manter várias tabelas em diferentes locais da memória e usar a mesma rotina para pesquisar qualquer uma delas. Se sua rotina encontrar o endereço-base da tabela indiretamente, por meio de um indicador, poderá ser usada com qualquer tabela, desde que se modifique o conteúdo do indicador antes de se chamar a rotina.

Em geral, os programas requerem combinações desses modos de endereçamento. A instrução LDA do 6502, por exemplo, pode ser usada nos seguintes modos:

Código de operação	Operando	Modo
LDA	#$34	Imediato
LDA	$A2	Direto, página zero
LDA	$967F	Direto, absoluto
LDA	$A2,X	Página zero, indexado com X
LDA	$967F,X	Absoluto, indexado com X
LDA	$967F,Y	Absoluto, indexado com Y
LDA	($A2,X)	Indireto, pré-indexado com X
LDA	($A2),Y	Indireto, pós-indexado com Y

Exemplificamos com uma instrução do 6502 porque mostra com clareza as combinações de diversos modos de endereçamento. Essa tabela pode, de início, parecer um pouco confusa, mas, na verdade, o uso de vários modos torna claro seu significado; até agora usamos LDA e ADC em dois modos — imediato e absoluto — sem confusão.

A tabela responde a uma pergunta natural: como saber qual o modo de endereçamento de uma instrução quando o código mnemônico é o mesmo em todos os casos? Vemos que o formato do operando difere em cada modo, e a única ambigüidade possível é quanto a uma instrução como LDA SYMB1 — ela está no modo absoluto ou página zero? Um programa assemblador resolve esse problema. Mas, se você quiser compilar o programa manualmente, terá de determinar se SYMB1 foi definida como um valor de 1 ou de 2 bytes.

De modo geral, uma vez que se comece a usar um programa assemblador, pode-se esquecer dos códigos de operação e do número de bytes por instrução e concentrar-se em aprender as técnicas de programação do ASSEMBLY. É importante entender como funciona o código de máquina, mas a linguagem ASSEMBLY, usada em conjunto com um bom programa assemblador, é um método de programação muito melhor, que combina a potência do código de máquina com muitos recursos das linguagens de alto nível.

Eis dois programinhas úteis em ASSEMBLY prontos para execução.

```
; CONVHA — CONVERTE BYTES HEXA EM CARACTERES ASCII
; SEQUENCIA DA CHAMADA

; LXI     D,<OP1>    CARREGA O ENDERECO DE DESTINO
; LXI     H,<OP2>    CARREGA O ENDERECO-FONTE
; MVI     C,<CONT>   CONTADOR DE BYTES HEXA (1-255)
; CALL    CONVHA
CONVHA:   MOV    A,M       ; CARREGA DOIS DIGITOS HEXA DE 4 BITS
          RAR              ; EXAMINA O PRIMEIRO DA ESQUERDA
          RAR              ;
          RAR              ;
          RAR              ;
          CALL   COHAD     ;
          MOV    A,M       ; CARREGA-OS NOVAMENTE E
          CALL   COHAD     ; EXAMINA O DIGITO DA DIREITA
          INX    H         ; INDICA O BYTE HEXA SEGUINTE
          DCR    C         ; CONT=CONT-1
          JNZ    CONVHA    ; CONTINUA O LOOP
          RET              ; RETORNA AO USUARIO
COHAD:    XCHG             ; TROCA O CONTEUDO DOS INDICADORES
          ANI    OFH       ; MASCARA ZERO NOS 4 BYTES DA ESQUERDA
          OPI    10        ; EXAMINA A FAIXA
          JNO    COHA1     ;
          ADI    D30H      ; FAIXA 0-9
          JMP    COHA2     ;
COHA1:    ADI    037H      ; NA FAIXA A-F
COHA2:    MOV    M,A       ; ARMAZENA O RESULTADO
          INX    H         ; INDICA O PROXIMO BYTE
          XCHG             ; DESTROCA OS INDICADORES
          RET              ; RETORNA

; TRANS: TRANSFERE DADOS DE UMA AREA (OP1) PARA OUTRA AREA (OP2)
; SEQUENCIA DA CHAMADA:

; LXI     D,<OP1>    ; CARREGA ENDERECO DE ORIGEM
; LXI     H,<OP2>    ; CARREGA ENDERECO DE DESTINO
; MVI     C,<CONT>   ; CARREGA TAMANHO DO CAMPO OP1

; CALL TRANS         ; TRANSFERE OS DADOS DE OP1

TRANS:    PUSH   H         ; SALVA ENDERECO DE OP2
TRANS1:   PUSH   D         ; SALVA ENDERECO DE OP1
          LDAX   D         ; ACESSA UM DADO DE OP1
          MVI    D,0       ; ZERA O REGISTRADOR D
          MOV    E,A       ; CARREGA O ACUMULADOR NO REGISTRADOR
                           ;   E
          DAD    D         ; ENDERECO DE ENTRADA DE OP2
          MOV    A,M       ; ACESSA UM DADO DE OP1
          POP    D         ; RESTAURA ENDERECO DE OP1
          POP    H         ; RESTAURA ENDERECO DE OP2
          PUSH   H         ;
          STAX   D         ;
          INX    D         ; INCREMENTA O INDICADOR DE OP1
          DCR    C         ; INCREMENTA A CONTAGEM
          JNZ    TRANS1    ; LOOP ATE O FINAL
          POP    H         ;
          RET              ; RETORNA
```

LABO

A Labo, indústria brasileira de mínis, produz também micros, terminais bancários e sistemas aplicativos. Modulares e compatíveis entre si, seus equipamentos vêm conquistando crescente fatia do mercado.

Criada em 1961, a Labo Eletrônica S.A. iniciou suas atividades fabricando instrumentos eletrônicos de alta precisão. Em 1979 entrou na área da informática, firmando um acordo de transferência integral de tecnologia com a empresa alemã Nixdorf Computer A.G. Esta, a maior fábrica européia de computadores e detentora de tecnologia avançada, já instalou mais de 110.000 sistemas em todo o mundo. Em 1983, a Labo participou com 27,7% das unidades comercializadas no Brasil, alcançando em 1984 a marca dos 1.500 equipamentos instalados.

A linha de produção da empresa inclui os minicomputadores da série Labo 8000, com vários modelos totalmente compatíveis em hardware e software; os supermínis, de arquitetura modular e atualizada; e os microcomputadores, que constituem a série Labo 8200. Fornece, ainda, sistemas bancários on line e off line e terminais de venda para lojas e supermercados.

Modulares e compatíveis entre si, os equipamentos Labo expandem-se e integram-se facilmente. Seus micros e mínis comunicam-se em rede. Mantendo os investimentos já realizados em hardware, software e treinamento, converte-se o míni 8034 num 8038, com um processador central duas vezes mais rápido e um poderoso software básico.

O micro Labo 8221, concebido para operar em configuração multiterminal, dispõe de uma capacidade de memória superior a qualquer outro equipamento de seu porte. Nele, o processador central, a memória e o controlador da unidade de discos estão montados numa única placa, o que simplifica sua manutenção. Trabalhando com três modos de operação hierarquizados — usuário, sistema e núcleo (kernel) —, proporciona uma proteção aos dados só encontrada nos sistemas de maior porte. Seu software básico é o SOL (Sistema Operacional Labo), que permite, operando com diferentes terminais, executar dois programas simultaneamente.

Também o minicomputador Labo 8043, o maior fabricado pela empresa, oferece total proteção a programas e dados: qualquer acesso às informações é protegido por um conjunto de procedimentos conhecidos apenas pelo pessoal autorizado. Outro valioso recurso desse míni está no módulo ASA (Aceleração de Software Aplicativo). Ele reduz o número de acessos aos discos, mantendo na memória, de forma dinâmica, os blocos mais lidos e prevendo por estatística os próximos blocos a serem acessados. Totalmente desenvolvido no Brasil, o ASA agiliza o trabalho do míni, equiparando-se, nos recursos, aos equipamentos de grande porte.

Os componentes utilizados pela Labo são rigorosamente testados em seu laboratório de controle de qualidade, onde uma câmara de simulação de envelhecimento descarta todos os elementos que apresentarem falhas.

Visando oferecer soluções imediatas aos problemas de seu cliente, a Labo dispõe de uma série de programas prontos, destinados a diversos segmentos do mercado: consórcios, hospitais, transportadoras, frotistas, administradoras de imóveis, operadoras de open market, agroindústrias etc. Todos os programas são protegidos por códigos e procedimentos que impedem o acesso de pessoas não autorizadas às informações.

Com filiais nas principais cidades do Brasil, a Labo dedica especial atenção à manutenção de seus equipamentos, incluindo suporte remoto por telediagnóstico — serviço inédito que, através das linhas telefônicas, detecta irregularidades nos sistemas e programas.

Entre esses programas destacam-se o SACIL — Sistemas Aplicativos Comerciais Integrados Labo e o PLANCOI — Planejamento e Controle Industrial, destinado, principalmente, a empresas de médio e grande porte.

Programas protegidos
O 8043, o maior dos mínis fabricados pela Labo, conta com recursos de proteção a programas e dados confidenciais. Dispõe também de um módulo ASA, que reduz os acessos aos discos, agilizando a execução.

PERFIL DO FUTURO

Um operário metálico

Sensores
Infravermelho, de luz etc.

Antena
Comunicações a distância

Unidade de discos
Para entrada e saída de dados e programas.

Força Mão-ferramenta

Saída impressa
Para recibos, programações, instruções

Monitor e teclado
O robô é um computador móvel. E, para ser usado como tal, os engenheiros de manutenção também precisariam de acesso ao software e aos programas de diagnóstico.

Carga/armazenamento

Sondas
Vários instrumentos de medição, canetas ópticas, sondas lógicas.

Alto-falante
Propicia saída de fala sintetizada

Microfone
Para reconhecimento de fala e som.

Pé
O robô se "senta" sobre sua base, gira as pernas e transforma os pés em garras, substituindo assim a mobilidade e força por uma enorme capacidade de manipulação.

Tempo futuro
Tudo indica que ainda teremos robôs não especializados, de aparência humana, a um custo capaz de torná-los os substitutos do trabalho humano semi-especializado. Mas eles precisarão de uma "inteligência" altamente desenvolvida que inclua um banco de dados de conhecimentos e habilidades técnicas, integração sensorial e software de aprendizado.

Entre os vários aspectos relativos aos modernos robôs estão suas limitações práticas. Mas as conquistas tecnológicas já apontam os possíveis desenvolvimentos futuros.

Os robôs reais estão muito longe daquelas máquinas pensantes concebidas pela ficção. Nossa imaginação nos condicionou a esperar deles determinadas qualidades: que sejam capazes de se mover à vontade; de ver, ouvir e sentir as coisas ao seu redor; de conversar conosco sobre filosofia e ciência ou, pelo menos, de comunicar-se de modo inteligente; e de manipular objetos e idéias tal como fazemos. Em outras palavras: nós os criamos à nossa imagem e semelhança. Assim, quando examinamos criticamente os robôs usados para fins comerciais, industriais ou amadorísticos, muitas vezes nos admiramos de como desempenham suas tarefas específicas, ao mesmo tempo que nos decepcionamos por não conseguirem fazer nada além disso. Partindo do que conhecemos hoje sobre projetos e implementação de robôs, o que podemos esperar — em termos realistas — dos robôs do futuro?

Movimento

É pouco provável que, num futuro próximo, os robôs venham a se locomover sobre algo parecido com as pernas humanas. Seria necessário dedicar muito tempo e espaço de processamento ao esforço de manutenção do equilíbrio, já que as articulações e a "musculatura" elétrica e hidráulica do robô não têm flexibilidade igual à dos seres humanos (graças à interação de músculos, tendões e cartilagens). Além disso, em muitas situações, o deslocamento sobre duas pernas seria um obstáculo.

Existem robôs experimentais de quatro e seis pernas, como insetos. Isso oferece uma interessante variação ao projeto dessas máquinas para determinadas utilizações. Mas, em operações militares, explorações interplanetárias e usos domésticos convencionais, as rodas se mostram mais práticas. No futuro, o movimento do robô será mais flexível, mas jamais poderá competir com a desenvoltura e naturalidade de um atleta em ação.

Robôs industriais e braços mecânicos são, na maioria, necessariamente fixos, com movimentos limitados a uma área de ação bem específica, já que foram projetados para desempenhar apenas uma ou duas tarefas. Mesmo que se modifique de maneira radical a estrutura das linhas de montagem, é provável que, ainda assim, robôs industriais com capacidade de se deslocar permaneçam relativamente confinados: seguirão linhas demarcadas, correrão sobre trilhos, deslizarão sob suportes aéreos. Pode ser que os aperfeiçoamentos na automação e no projeto de robôs tragam mudanças radicais nos métodos de produção industrial, mas não dá para prever quais serão tais mudanças.

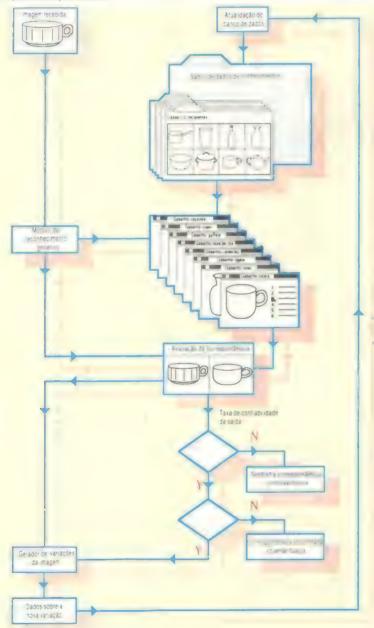
Um elemento-chave do movimento é a necessidade de o robô reagir ao ambiente. Isso implica equipá-lo com elementos sensores, devendo essa entrada de informações estar diretamente relacionada com sua capacidade de movimento.

Variações visuais
O conhecimento de um objeto pode ser armazenado como um "gabarito" do original, juntamente com dados sobre sua variação; cada variação pode ser aplicada ao gabarito para produzir uma imagem diferente, mas que ainda se encaixa numa definição do original. Um módulo de análise geral esquadrinha a imagem proveniente dos sensores do robô, possibilitando uma indicação sobre o tipo genérico do objeto. Todas as variações são então comparadas com a imagem recebida até que haja uma correspondência. A certeza estatística com que se obtém essa correspondência determina se as novas variações da imagem em relação ao gabarito apresentam-se tão profundas a ponto de exigirem uma nova instrução de variação de dados

Sensores

Podemos adaptar ao robô equipamentos sofisticados que ampliem sua capacidade de percepção em áreas novas e mesmo desconhecidas para o homem. Sensores de proximidade, detectores de movimento, dispositivos precisos de realimentação posicional descontínua, captadores de ruídos numa ampla gama de freqüências etc., tudo isso dá ao robô capacidade para coligir uma enorme variedade de dados. Bem mais que os seres humanos. Os sistemas visuais vêm adquirindo precisão cada vez maior, com melhoria crescente na resolução das imagens percebidas. As técnicas de síntese e de reconhecimento da voz, embora ainda em desenvolvimento, tendem a maior sofisticação e terão lugar importante no aperfeiçoamento da robótica.

Robôs utilitários, especialmente na indústria, continuarão a ser equipados apenas com os sensores necessários para execução de tarefas específicas. Braços soldadores, por exemplo, prescindem da capacidade de fala ou de realimentação visual. Eles executam suas tarefas com precisão e rapidez, tendo um mínimo de entrada sensorial, e as percepções irrelevantes talvez só viessem perturbar-lhes o desempenho.

Os robôs não especializados, projetados para simular processos do pensamento humano e aprender com a experiência, teriam de ser equipados com o maior número possível de sensores. O fundamental seria que pudessem investigar o ambiente com autonomia e assimilar as informações que coligissem. Os seres humanos apóiam-se numa combinação de informações visuais e auditivas para compreender a fala. Muitas vezes tendemos a ignorar essa síntese, principalmente quando pensamos em termos de projeto de robôs. Mas, para que eles possam se comunicar com o homem, compreender sua linguagem, em vez de apenas reconhecê-la, seria essencial conjugar visão e audição.

Hoje, o volume de dados que os robôs recebem e com o qual podem lidar é restringido pela quantidade de memória necessária para armazenar as entradas sensoriais e pelas limitações na capacidade e velocidade de processamento. Um robô é capaz de armazenar a imagem visual de um objeto — uma maçã, por exemplo — e vincular essa imagem a um nome. O armazenamento da imagem ocupa espaço na memória e, quanto maior a resolução visual, maior a quantidade de memória requerida para manter a imagem. Como as maçãs não são todas iguais, o robô deverá ter capacidade suficiente de memória para armazenar vários modelos representativos de imagens de maçãs. Ou, então, um algoritmo que identifique as variações e gire a imagem básica, de modo que a maçã seja reconhecida em qualquer posição. Mesmo com um nível mínimo de resolução (digamos, 256 pixels por imagem), o número de variações pode chegar à casa dos milhares.

Futuramente, as exigências de memória talvez venham a ser preenchidas pelos chips RAM de grande capacidade (chips de 1 Mbit já se encontram em desenvolvimento) e pelos chips RAM interativos, que armazenam dados "variacionais". Os dados não específicos mantidos nesses chips, sempre que necessário, podem ser chamados pelo processador, para elucidar imagens diferentes, quase como uma sub-rotina num programa em BASIC.

Afora isso, a verdadeira percepção sensorial do robô exigiria que os dados chegassem de várias fontes ao mesmo tempo e que fossem rapidamente executados. Os processadores existentes seriam incapazes de lidar com um volume tão grande de informações simultâneas. Assim, os dados logo começariam a se acumular, entrando em ordem de espera. Uma provável solução estaria no uso de dois ou mais processadores de alta velocidade e capacidade operando paralelamente. Nesse caso, um processador de controle funcionaria como gerenciador, distribuindo tarefas aos processadores inativos em algum ponto do sistema.

O problema, porém, não está apenas no hardware. Para compreender o objetivo do processamento, o robô precisará de um software muito complexo. Isto é, uma verdadeira "mente".

Mente

Como já vimos, duas são as direções principais da robótica. A que tem maiores possibilidades é a área de instrumentos inteligentes, ou robôs utilitários. Braços mecânicos e sistemas de fabricação automatizados que executam tarefas específicas só requerem um equipamento de software de controle cuidadosamente definido. O braço receberia então um conjunto de coordenadas e uma programação para executar uma seqüência de ações, sem precisar saber o que faz, onde está ou qualquer detalhe sobre o ambiente.

À medida que os robôs são solicitados a executar uma variedade mais ampla de tarefas, porém, estas têm de ser, necessariamente, menos definidas. Se eles tiverem de se movimentar numa sala onde a posição dos móveis se modifica todos os dias, precisarão não apenas coligir e processar informações, mas também incorporar novos elementos em sua percepção do mundo. Seria imprescindível um software de controle operacional expansível segundo a solicitação.

A criação de uma mente mecânica suscita importantes questões, até agora sem respostas, a respeito de como o homem pensa e aprende. Por exemplo: o que de fato sabe uma criança ao nascer? O adulto é produto do meio ou de sua hereditariedade? E qual a relação entre esses parâmetros? O homem parte de um conjunto de estruturas internas que o ajudam a aprender a linguagem, a matemática e a adquirir sensibilidade estética? Se é assim, como funcionam tais estruturas? Para responder a essas perguntas, precisamos estudar o cérebro humano. No futuro, talvez os robôs nos ajudem no desenvolvimento de uma compreensão maior sobre nós mesmos. Embora essas questões não sejam totalmente teóricas — como experiências nesse sentido vêm atestando —, a idéia de um robô pensante ainda está bastante longe.

Comboio!
As técnicas da robótica exercerão maior impacto sobre a sociedade quando forem incorporadas às ferramentas especializadas de baixo nível tais como guindastes, caminhões basculantes, veículos para entregas locais e transporte pesado.
Na figura, um robô-basculante carrega com terra uma série de robôs-caminhões. Depois de carregado, cada robô-caminhão se move de modo semi-inteligente até um ponto de encontro e se engata a uma fila de outros. O comboio se move então pela força de cada robô-caminhão, mas controlado por um homem dirigindo o primeiro veículo. A união da capacidade humana de decisão e comando com a força e a inteligência de finalidade única dos robôs é a maneira mais barata e lucrativa de unir os recursos de hoje com a tecnologia de amanhã.

MODELOS EM PERSPECTIVA

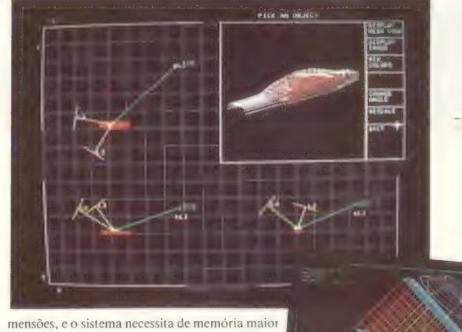
O emprego do computador na geração de gráficos é um dos segmentos de tecnologia mais avançada na informática. Vamos ver os recursos oferecidos aos usuários por um dos mais sofisticados sistemas: o CAD.

A diferença entre os desenhos elementares em computador e os obtidos com o sistema CAD se evidencia no significado da própria sigla: Computer Aided Design ("projeto com auxílio do computador"). Em sua memória se aloja uma perspectiva real do objeto, e não um simples desenho. Os sistemas CAD mais sofisticados podem projetar um automóvel incorporando cada uma de suas peças e, em seguida, simular tensões e deformações, assegurando-se assim de que ele não se fará em pedaços quando em uso.

Na maioria dos CADs, os desenhos são elaborados de modo diferente dos sistemas gráficos comuns, construindo-se as imagens por meio de pixels (células pictóricas); o CAD utiliza gráficos vetoriais bem mais precisos. Nos gráficos com pixels, uma linha consiste numa série de pontos na tela; já no sistema vetorial, é entendida como a menor distância entre as coordenadas das duas extremidades. O desenho baseia-se na descrição matemática das formas, e a maior precisão permite ao processador trabalhar com resolução muito maior que a do monitor. Tudo isso assegurou ao CAD, sistema criado em 1963, uma utilização crescente nas indústrias mecânica e eletrônica, na engenharia e na arquitetura, na topografia e nas publicações técnicas.

Os CADs mais simples criam desenhos bidimensionais. Algumas vezes podem sugerir uma terceira dimensão — que normalmente se chama de desenho em 2 1/2 D. A "meia" dimensão extra atende às necessidades de um escritório de arquitetura, por exemplo, permitindo a visualização das elevações, planos de base e a análise da declividade de um terreno.

Algumas companhias que projetam circuitos eletrônicos usam sistemas bidimensionais de maior complexidade. Em geral, na eletrônica, o CAD encontra-se associado a programas de verificação. Depois que um projetista de chips mapeia os milhões de circuitos de um microprocessador, o CAD faz a checagem do funcionamento desses circuitos.

O uso mais complexo do CAD prende-se a projetos industriais, que chegam a integrar milhares de peças. Os desenhos requerem três dimensões, e o sistema necessita de memória maior e processamento mais rápido para tratar a imagem. Nesses casos, o projetista mapeia primeiro cada uma das peças — todas numa escala específica — e as armazena separadamente. À medida que o trabalho avança, os diferentes elementos podem ser recuperados do banco de dados e incorporados ao desenho. CADs com essas características são essenciais às grandes empresas automobilísticas. Por exemplo, a Volkswagen do Brasil utilizou as estações gráficas desenvolvidas pela Intergraph americana — fornecedora de tecnologia à Sisgraph brasileira — durante o projeto do automóvel Santana.

Desenhos em luz e sombra

Num sistema CAD, a mais simples forma de desenho tridimensional, a chamada Wireframe, mostra somente os contornos do objeto, sem imagens das superfícies. Sistemas mais sofisticados podem produzir imagens tridimensionais coloridas, com até 256 cores diferentes na tela ao mesmo tempo. Nesses sistemas, é possível produzir imagens de superfícies multicoloridas e sombreadas, opacas ou translúcidas, mostrando os efeitos de múltiplas fontes de luz.

Os usuários de CADs têm a seu dispor uma ampla gama de recursos. Destacamos a seguir os mais importantes.

Dimensionamento. Terminado o desenho, o sistema pode acrescentar automaticamente todas as medidas numéricas, deixando o projeto pronto para apresentação.

Hachuras. Processo de sombreamento automático, usado em desenhos técnicos quando se pretende destacar determinadas partes do projeto.

Superposição. Os projetos podem ser elaborados em camadas. Por exemplo, um arquiteto colocaria a planta de uma casa numa camada e a das canalizações de água e esgotos em outra. As plantas apareceriam em cópias separadas ou na mesma cópia.

Segmentos. Os elementos que compõem o desenho são armazenados num "catálogo" de desenhos prontos, que o projetista pode chamar a qualquer momento.

Expansão/condensação. Esse recurso dá ao projetista a possibilidade de alongar ou encurtar linhas — como se fossem um elástico — para colocá-las onde for necessário.

Ajuste de escala. Permite ao desenhista projetar no tamanho que queira e em seguida armazenar as informações na escala do restante do desenho.

Zoom e pan. Termos tomados por empréstimo ao cinema. No zoom, ampliamos uma parte do desenho até que encha a tela; com o pan, ao contrário, a afastamos, a fim de visualizarmos o conjunto do desenho.

Dispositivos de entrada

Os usuários de CADs geralmente escolhem os dispositivos de entrada conforme o trabalho a realizar. Entre os mais familiares estão o tablete digitalizador, a caneta óptica, o rastreador (para leitura automática de textos impressos em papel) e o mouse.

Usam-se os processadores de imagens para a introdução de fotografias e desenhos complexos. Eles fotografam a figura a ser introduzida e decompõem a foto em elementos reticulados, que formam uma imagem na tela.

O teclado é um meio eficaz de introduzir as coordenadas dos gráficos vetoriais. Juntamente com o mouse, é um dos componentes do Interpro 32, a estação gráfica produzida pela Sisgraph, com microprocessador de 32 bits, que pode ser utilizada em separado ou ligada a redes de computadores já instaladas.

Também se usam joysticks, só que bem mais sofisticados que os dos jogos eletrônicos. O Bitstik, da Robocom, tem potenciômetros que medem o movimento do controlador com precisão de até seis casas decimais. O joystick comanda um cursor na tela e, com os botões, selecionam-se funções no menu. A extremidade do bastão pode ser ajustada para ampliar um detalhe ou introduzi-lo no desenho completo.

Desenho de precisão por meio do Bitstik

O CAD no micro doméstico
O sistema gráfico CAD torna-se uma ferramenta cada vez mais confiável, para criação e testes de projetos sem a necessidade de se construírem modelos, sempre caros. Muitos sistemas hoje disponíveis destinam-se a micros domésticos de pequena capacidade, embora exijam periféricos e equipamentos adicionais para resultados a nível profissional.

Para funcionar com eficácia, o CAD exige muito software adicional. O enorme volume de informações a processar requer uma grande capacidade de memória. O usuário precisa de um controlador de largura dupla e uma placa de memória, e placas adicionais de memória caso se acrescentem cores. Para aceitar informações do computador MTX, é necessário um conversor A/D "flash" simples (monocromático) ou de três canais (em cores). Um sistema totalmente equipado armazena dados, vindos da câmara de vídeo, à velocidade de 7,2 Mbytes por segundo — e tal potência custa caro.

Já o sistema Bitstik é bem mais acessível. O pacote foi desenvolvido para o BBC Micro e permite a produção de desenhos técnicos de qualidade profissional. Consiste num controlador — uma espécie de joystick de alta precisão —, disco de sistemas, ROM de aplicativos com 8 Kbytes e disco-biblioteca contendo várias famílias de letras e símbolos comuns. Entretanto, para rodar o pacote, deve-se associar o BBC Micro a um segundo processador 6502 e a duas unidades de disquetes de oitenta trilhas. E, para obter o máximo do sistema, o usuário precisará de um monitor colorido e de um traçador, também em cores.

Mas o investimento parece valer a pena: segundo os fabricantes, com esses equipamentos, mesmo pessoas sem inclinação para o desenho produzem gráficos de padrão profissional. A base do sistema é a biblioteca de formatos mantida em disco e as "primitivas" — linhas e arcos armazenados em ROM. A partir das primitivas, elaboram-se conjuntos de diagramas cada vez mais complexos, que então podem ser incorporados a desenhos maiores. A parte principal do sistema está no controlador Bitstik, que permite o posicionamento das primitivas com absoluta precisão.

A LUZ FANTÁSTICA

Recentes avanços no campo da computação óptica possibilitam o uso de feixes luminosos para transmissão de informações a uma velocidade milhares de vezes superior à dos computadores eletrônicos.

Condutor de fibras
O equivalente óptico do fio elétrico é o cabo de fibras. Permite "encurvar" os sinais luminosos e, assim, direcioná-los à vontade. As fibras que preenchem o cabo contêm uma substância transparente que "conduz" a luz na direção da fibra.

Quando pensamos em "computador", em geral visualizamos um aparelho que se liga na tomada, baseado em interruptores binários eletrônicos que armazenam e processam informações. Entretanto, é errado concluir que esse seja o único tipo possível de computador. Teoricamente, um computador pode basear-se em qualquer método de armazenamento de informações: carga elétrica, engrenagens, orifícios num papel, contas, pedras, ou qualquer outro meio.

O semicondutor eletrônico constitui o melhor tipo de armazenamento inventado até hoje, pois consome pouquíssima energia, tem tamanho microscópico e, sobretudo, é rápido. A velocidade de operação do computador fica limitada apenas pelo tempo que seus interruptores binários necessitam para passar de uma posição a outra, e pelo tempo de transmissão de informações entre eles. Circuitos baseados em fluxos de elétrons mostram-se muito rápidos na execução dessas funções. Mas já são possíveis, teórica e tecnicamente, circuitos ainda mais rápidos, baseados em feixes de luz. Existem pesquisas em computação óptica e em disciplinas a ela vinculadas, como a "lógica fotônica".

Muitos componentes necessários a um computador óptico já se encontram disponíveis no mercado: utilizam-se fibras ópticas, por exemplo, para transportar sinais telefônicos e de televisão. Acham-se desenvolvidos guias de ondas ópticos, equivalentes aos condutores metálicos empregados nos painéis de circuitos impressos. Também os discos ópticos já são realidade. O componente fundamental, que seria o equivalente óptico do interruptor binário eletrônico, encontra-se em fase de testes.

Pesquisam-se dois tipos desses interruptores: os ópticos e os eletroópticos. Vale a pena examinar cada um, começando pelo eletroóptico, já que, num primeiro momento, a tendência será associar componentes ópticos e eletrônicos. Ambos se valem da interferência luminosa ocasionada por alterações controladas no índice de refração de um material óptico não linear, em geral um composto de gálio ou lítio. Esse índice expressa a proporção entre as velocidades da luz no vácuo e naquele material. Por exemplo, se o índice de refração de um determinado tipo de vidro é de 1,5, e dado que a luz viaja no vácuo a uma velocidade de cerca de 300.000 km/s (quilômetros por segundo), sua velocidade no vidro será de cerca de 200.000 km/s.

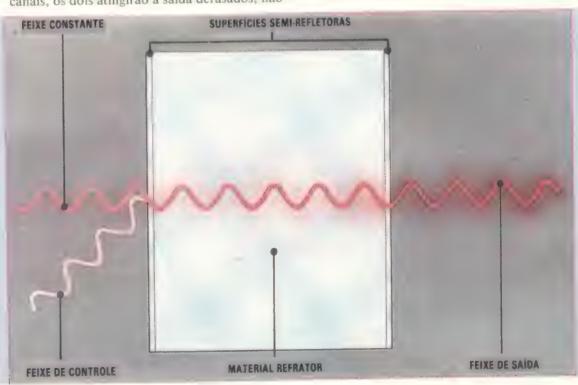
Programa luminoso
Outro componente essencial para a concretização do computador óptico é este dispositivo semicondutor, que emite uma faixa de freqüências de luz pura. Isso será fundamental no tocante às interferências nas comunicações, onde se requer uma padronização na freqüência e no comprimento das ondas luminosas.

O interruptor binário eletroóptico baseia-se no interferômetro Mach Zehnder, dispositivo cujo princípio de funcionamento é muito simples. Um feixe luminoso que o atinge divide-se em dois, passa por canais independentes, feitos de material não linear, sendo então recombinado. Caso não seja aplicada uma corrente elétrica a um dos canais do interferômetro, os dois raios chegam à saída em fase e geram um forte feixe — o estado "ligado". Se alterarmos, por meio de um sinal elétrico, o índice de refração de um dos canais, os dois atingirão a saída defasados, não se recombinando, pois suas velocidades serão diferentes. Resultará um feixe de saída muito mais fraco — o estado "desligado".

Esse tipo de interruptor constituiria a base de um circuito integrado óptico. Seria mais rápido que um equivalente eletrônico, pois a luz transporta informações a uma velocidade muito maior que a dos sinais eletrônicos (cerca de 10.000 vezes mais rápida, devido à freqüência mais alta das ondas luminosas). Apesar desse desempenho superior, a velocidade do circuito misto ainda fica limitada pela realimentação elétrica, necessária para operar os interruptores. Podem-se obter velocidades ainda maiores usando a luz (um feixe fraco controla outro mais forte).

Esse tipo de estrutura é análoga à do transistor eletrônico. Um protótipo de interruptor inteiramente óptico, chamado transfasor, já foi construído por uma equipe da Universidade de Heriot Watt, em Edimburgo, na Escócia. Esse dispositivo liga e desliga em cerca de 1 picossegundo (1 ps = 1 trilionésimo de segundo, ou seja, 1 ps = 10^{-12} s). Em comparação, o mais rápido transistor eletrônico possui uma velocidade de ligação de 1 nanossegundo (1 ns = 1 bilionésimo de segundo, ou seja, 1 ns = 10^{-9} s). Isso significa que um computador inteiramente óptico operaria a velocidades mil vezes superiores a um eletrônico.

O transfasor desenvolvido na Universidade de Heriot Watt baseia-se num interferômetro inventado pelos físicos franceses Charles Fabry e Alfred Perot em 1896. Nele, coloca-se um material não linear entre duas superfícies refletoras. A luz que atravessa uma delas ricocheteia na camada intermediária antes de sair pela superfície oposta. As ondas de luz interceptadas interagem de uma forma determinada pelo índice de refração

Sobre o feixe luminoso
O elemento básico do computador eletrônico é o transistor; seu equivalente óptico chama-se transfasor. Um feixe constante entra por uma das extremidades, juntamente com um feixe de controle; ambos interferem em conjunto no interior do dispositivo. Uma pequena variação na intensidade do feixe do controle produzirá notável mudança na saída luminosa. Desse modo, o transfasor pode ser usado como interruptor lógico.

do material não linear, que pode ser ajustado por um feixe de controle. Se o índice de refração for tal que as ondas luminosas fiquem em fase e se reforcem mutuamente, o resultado será um feixe luminoso forte; se ficarem fora de fase, o feixe transmitido resultará fraco. Os dois estados de transmissão alternativos correspondem às posições "ligado" e "desligado".

Vantagens teóricas

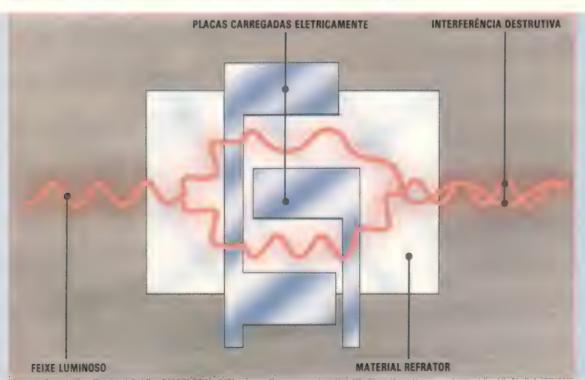
O sucesso do computador óptico será decidido, em última análise, pelos limites teóricos e técnicos das máquinas convencionais. Em tese ele é superior a seus equivalentes eletrônicos em diversos aspectos, começando pela velocidade de comutação. Para que um transistor funcione como interruptor, uma corrente de elétrons deve atravessar sua base. Há limites para a velocidade de movimento dos elétrons num semicondutor; bem como para a espessura mínima de um transistor (quanto mais fina sua base, mais curto o trajeto dos elétrons). Um computador óptico, transmitindo à velocidade da luz, é inerentemente muito mais rápido.

As máquinas convencionais também são limitadas pelo chamado "engarrafamento de Von Neumann". Característico da arquitetura básica de quase todos os computadores, diz respeito ao modo serial de acesso às informações na memória (uma palavra por vez). Um computador óptico acessa a memória de modo paralelo, pois os feixes luminosos não precisam ser isolados, como ocorre com os sinais elétricos. Diversos feixes separados, assim, correm pelo mesmo transistor óptico e operam simultaneamente.

A terceira limitação do computador convencional consiste no pequeno espaço disponível para conectar fios num chip. Poucas conexões são possíveis, devido ao problema do leque de pinos — limite quanto ao número de pinos que podem ser montados numa fina camada de silício. O uso de técnicas tridimensionais holográficas (ópticas) possibilitaria mais liberdade no sistema de fiação e permitiria modificar instantaneamente a configuração das conexões.

Finalmente, há vantagens práticas na interconexão dos computadores ópticos e dos periféricos. Já se usam redes de fibra óptica para ligar dispositivos eletrônicos, sendo as mensagens luminosas extremamente rápidas, além de se poderem transmitir várias delas na mesma fibra sem interferência mútua. Essas redes conectam vários computadores ópticos sem necessidade das lentas interfaces optoelétricas.

Em vista dessas vantagens, para quando podemos esperar o surgimento de computadores ópticos práticos e financeiramente acessíveis? Eis a resposta: eles ainda demorarão algum tempo. As vantagens da óptica sobre a eletrônica são maiores na teoria do que na prática. Os crescentes avanços na miniaturização dos chips significam que, pelo menos a curto prazo, os computadores ópticos estão perseguindo um objetivo cada vez mais alto. O número de circuitos num único chip vem dobrando a cada dezoito meses. Chips com 10 ou 100 milhões de circuitos estão previstos para o ano 2000. A chamada LSI (Large Scale Integration, "integração em larga escala"), transformou-se em integração a altíssima (e ultra) escala. O "processador paralelo" é hoje "maciçamente paralelo".

A tecnologia eletrônica de alta velocidade encontra-se em contínuo desenvolvimento, sob dura competição. Visto por esse prisma, o computador óptico a custo acessível pode estar a anos-luz de distância.

Fase a fase
O interferômetro Mach Zehnder permite o funcionamento conjunto de dispositivos eletrônicos e ópticos. A luz que entra por uma das extremidades divide-se em dois feixes. Após a entrada dos feixes no dispositivo, ajusta-se o índice de refração em torno de um deles. A velocidade da luz diminui, ficando então defasada em relação ao outro feixe. Quando os dois feixes convergem, na saída, as fases terão uma interferência destrutiva, resulta um "desligado", que pode ser considerado como um zero lógico.

SPECTRAVIDEO 318

O micro Spectravideo incorpora muitos elementos do padrão MSX, planejado por japoneses e americanos. O ponto alto desse equipamento, além de seu baixo custo, está nos recursos de expansão que oferece.

O Spectravideo 318 é um microcomputador de baixo custo que possui a mesma qualidade do BBC Micro e do Commodore 64. Incorpora sintetizador de som, gráficos de alta resolução com sprites, joystick embutido e entrada para cartucho. O teclado tem muito em comum com o do MSX e design semelhante ao do Sinclair Spectrum. Como este, suas teclas também possuem toque suavizado, e a distância entre elas torna sua operação muito mais agradável.

As teclas de comando do cursor transformaram-se num joystick embutido. Um disco substitui as costumeiras quatro flechas: tocando o disco na área apropriada, movimenta-se o cursor para cima, para baixo, para a esquerda ou para a direita. Mas pode-se encaixar, em lugar do disco, um joystick, que, ao ser acionado, produz o mesmo efeito das teclas; também é empregado na correção de um programa, posicionando o cursor sobre o erro a corrigir.

O BASIC do 318 é o mais atual de uma linha de interpretadores Microsoft. Essa versão aproxima-se do BASIC MSX, uma extensão do BASIC GW, usado em computadores como o IBM PC. É fácil elaborar programas nesse BASIC, que conta ainda com recursos genéricos, como a renumeração das linhas ou sua numeração automática. Embora não possua os novos comandos "estruturados", como WHILE-WEND e REPEAT-UNTIL, presentes em muitas das novas versões do BASIC, possui o IF-THEN-ELSE, que ajuda a escrever programas claros e eficientes. Esse BASIC tira o máximo proveito da capacidade gráfica do 318.

A tela possui razoável resolução, 256 x 192 pixels e dezesseis cores, embora pequenos grupos de pixels compartilhem as mesmas cores. Esses recursos podem parecer pobres se comparados aos de outros micros, mas é o nível de controle sobre o vídeo que possibilita bons gráficos e não

Teclado do Spectravideo
O teclado do 318 se aproxima muito das especificações MSX. O estilo assemelha-se ao do Spectrum, mas o espaçamento e a qualidade das teclas tornam perfeita sua utilização. É um teclado muito bem equipado com todas as teclas necessárias: [Tab], [Control], [Escape] e [Backspace], mais cinco para funções (cada uma duplicada), uma [Stop], outra [Select], e um conjunto para edição [Ins], [Del], e [Copy]. Um conjunto de formas gráficas (blocos, pontos e linhas) fica disponível ao se pressionar uma tecla alfabética junto com a [Left GRPH] ou com a [Right GRPH]. As formas que cada tecla produz são nitidamente delineadas no teclado.

apenas as especificações de tela. O 318 inclui todas as extensões gráficas do BASIC da Microsoft, como comandos para traçar pontos, retas, retângulos, círculos, arcos e elipses. Um comando PAINT preenche com uma cor qualquer forma geométrica fechada. As instruções especiais VPOKE e VPEEK permitem um controle ainda maior para ler e gravar diretamente na memória de tela. Há também uma minilinguagem gráfica, usada com o comando DRAW para criar formas e desenhos complexos.

Pode-se "pegar" qualquer retângulo no vídeo com um comando GET, armazená-lo numa matriz em BASIC, e substituí-lo na tela com um PUT. Isso facilita desenhar padrões geométricos e produzir animação simples e efeitos especiais, como a inversão de uma imagem. O toque final está nos sprites. O chip 9929 do monitor permite ao usuário desenhar suas próprias formas animadas: pessoas, invasores do espaço, mísseis.

A instrução ON SPRITE GOSUB permite criar uma "rotina de colisão": o programa roda normalmente, mas, se dois sprites colidem, desvia-se para essa rotina, que pode ser usada, por exemplo, para detectar o choque entre um míssil e uma astronave. Dessa forma, o programador não precisa checar continuamente todos os eventos como possibilidade de ocorrer; e os programas ficam mais fáceis de elaborar e rodam muito mais depressa. Rotinas como essa também estão disponíveis para as teclas de funções e para as que movem o cursor, entre outras.

Um bom som é essencial, e o chip sonoro do 318 possui três vozes e toda uma gama de efeitos especiais. O som pode ser reproduzido por televisão, o que permite um adequado controle de volume.

Há no 318 uma seleção de interfaces: duas saídas para joystick, uma entrada para cartucho, uma para cassete e um conector de expansão. Existe no mercado uma variedade de acessórios interessantes, mas costumam ser caros. Para conectar um deles, necessita-se de um miniexpansor, que permite memória extra, em geral uma expansão de 16 ou 64 Kbytes. Uma expansão ainda maior é conseguida mediante o superexpansor, que pode conectar até sete acessórios, utilizando um sistema de slots semelhante ao do Apple II. Incorporam-se mais memória, interfaces para impressora, unidades de discos e modems.

Se você for um entusiasta por jogos, poderá optar por um acessório muito interessante, o adaptador de jogos Coleco, que possibilita ao Spectravideo rodar cartuchos de videojogos dessa marca. Mas esse é um meio comparativamente dispendioso de rodar software de jogos.

O 318 requer um gravador cassete de sua própria marca, característica encontrada em núme-

Joystick do Spectravideo
Substitui as usuais teclas de movimentação do cursor. Sem o cabo, pode-se tocar um dos lados do disco para subir ou descer o cursor, ou movê-lo para a direita ou para a esquerda. O joystick tem o mesmo efeito, mas, empurrado em duas direções ao mesmo tempo, possibilita o movimento em diagonal, recurso impossível com as quatro teclas tradicionais.

Entrada para cartucho
Os cartuchos se encaixam neste conector, firmemente montado na placa principal.

ROM
O BASIC reside em duas ROMs de 16 Kbytes.

Conexão de expansões
Conectam-se aqui diversas expanções opcionais.

CPU
O processador utilizado é o conhecido Z80.

Saída para monitor
Acopla um modulador separado para acionar um aparelho de tevê. Isso permite usar o 318 com televisores de diferentes padrões, bastando selecionar o modulador apropriado.

Saída para cassete
Interface para gravador cassete de uso exclusivo do Spectravideo.

Versão aprimorada
Uma atraente alternativa ao 318 é o 328, a versão mais sofisticada, com teclado móvel, 80 Kbytes de RAM e software processador de texto residente. Tem por objetivo iniciar os que pretendem familiarizar-se com equipamentos profissionais.

Joystick incorporado
Um disco com cabo de joystick substitui as convencionais teclas do cursor.

Controlador de E/S
Um chip 8255 controla as interfaces de entrada e saída, tais como o joystick.

Som
Um chip 8910 gera três diferentes vozes.

Vídeo
O chip de vídeo 9929, encapsulado num dissipador térmico.

RAM
Dezesseis chips fornecem 32 Kbytes de RAM.

SPECTRAVIDEO 318

CPU

Z80

MEMÓRIA

32 Kbytes de RAM, dos quais cerca de 12 Kbytes disponíveis para programas em BASIC; 32 Kbytes de ROM.

TELA

Modo texto: 24 linhas x 40 colunas. Modo gráfico: alta resolução, 256 x 92 pixels, dezesseis cores e recursos para desenhar sprites. Há também uma opção de 80 colunas.

INTERFACES

Conector de expansões entrada para cartucho, duas saídas para joystick

TECLADO

Com atenuador de pressão; teclas de função e de edição; joystick incorporado, para movimento do cursor

LINGUAGENS DISPONÍVEIS

BASIC.

DOCUMENTAÇÃO

Reduzida e falha mas a chegada de outros micros MSX deve encorajar publicações independentes.

ro muito pequeno de computadores. Embora isso aumente a confiabilidade do sistema e a velocidade de armazenamento e de leitura, acaba encarecendo o equipamento.

Uma das características mais atraentes do Spectravideo reside em sua capacidade de expansão. Com a adição do superexpansor para mais 64 Kbytes de RAM, placa de 80 colunas e uma unidade de discos, a máquina se converte num pequeno computador comercial CP/M, com acesso a software profissional. Mas ainda é difícil encontrar programas compatíveis com seu formato de disquete, problema que persistirá até que o equipamento deixe de ser uma novidade e se popularize.

SOB MEDIDA

Adaptando o ambiente e o instrumental ao trabalho humano, a ergonomia tem por objeto projetar sistemas eficientes e que, ao mesmo tempo, preservem a saúde e o conforto de seus usuários.

A chegada dos microcomputadores ao lar e aos locais de trabalho torna as implicações ergonômicas importantes para todos nós. As cores e os sons constituem bons exemplos de como o ambiente afeta o desempenho das pessoas no trabalho. Algumas centrais telefônicas transmitem música para quem está aguardando na linha, a fim de amenizar a espera. Uma tela de vídeo verde ou âmbar é considerada pela maioria dos usuários agradável à vista e repousante. Já o branco e preto convencional parece cansativo e desgastante após algumas horas de uso.

Diferentes combinações de cores provocam reações bastante diversas: geralmente se considera azul sobre amarelo uma combinação atraente, enquanto azul sobre verde é impopular. A cor de uma sala pode afetar a disposição de seus ocupantes: tons amarelados são tidos como "alegres"; o azul e o verde, como "relaxantes"; e o marrom e o cinza, como "tristes".

Tais diferenças podem parecer pouco importantes ou mesmo irrelevantes para o proprietário de um micro. Mas os ergonomistas demonstraram que as pessoas, quando não estão satisfeitas com seu ambiente de trabalho (muitas vezes sem saberem que é o esquema de cores ou o ruído de fundo que as está afetando), tendem a sentir mais fadiga ocular, dores na coluna e têm maior probabilidade de faltar por motivo de doença. É muito comum usuários de computadores sofrerem por causa de esquemas de trabalho mal estruturados e salas mal divididas. Um importante estudo ergonômico, num escritório em que os operadores trabalhavam em terminais dispostos junto às paredes da sala, relacionou a baixa eficiência e o alto índice de erros ao isolamento social e desgaste causados pela disposição da sala. Quando os funcionários foram colocados frente a frente em terminais de baixa altura dispostos no meio da sala, o ambiente ficou mais agradável e a qualidade do trabalho melhorou sensivelmente.

A adoção de computadores resultou em menor especialização em alguns empregos, tornando o trabalho das pessoas desestimulante, rotineiro e pouco interessante. Hoje se reconhece como parte do trabalho do analista de sistemas garantir que as tarefas atribuídas às pessoas após a instalação de um computador sejam gratificantes e suficientemente estimulante. E, nessa análise, é necessária a orientação especializada de um ergonomista.

Num projeto de sistema ergonômico, o operador humano é um componente importante, com suas próprias características e tolerâncias operacionais. Reflexões de ordem tanto humanitária como financeira reconhecem que os sentimentos, percepções e hábitos das pessoas são no mínimo tão importantes para a eficiência de um sistema quanto a capacidade de seu bus de endereço ou a velocidade de seu clock.

O programador também pode se beneficiar com o estudo dos problemas e exigências de seu trabalho. A programação exige cuidado constante, método lógico e muita atenção aos detalhes, embora poucas pessoas apresentem essas características e a maioria resista à sua imposição. Essa é uma das razões por que não se podem garantir programas isentos de erro.

A aplicação da ergonomia na elaboração de novas linguagens de programação e métodos de desenvolvimento de sistemas é uma técnica fascinante, oferecendo muitas vantagens aos programadores e seus empregadores. Os psicólogos já dedicaram anos ao estudo das pessoas enquanto sistemas de processamento de informações, tendo hoje idéias claras (embora ainda incompletas) sobre estruturas da memória, velocidade e habilidade cerebral. Essas informações podem auxiliar no desenvolvimento de controles de pro-

As teclas do sucesso
A interação do usuário com o computador começa pelo teclado, que já foi submetido a muitos aperfeiçoamentos ergonômicos — desde teclas côncavas e teclados destacados, inclináveis, até visores de cristal líquido. Mas seu maior defeito — a ineficiência da disposição tipo QWERTY — parece insolucionável, pois a maioria das pessoas reluta em aprender a datilografar segundo novos padrões. Tais sentimentos de resistência a mudanças e o aparente irracionalismo constituem muitas vezes o principal obstáculo à engenharia dos fatores humanos.

gramas e na estruturação de dados com os quais as pessoas se sintam à vontade e que possam usar naturalmente ou com pouco treinamento.

Considerações de ordem psicológica há muito foram incorporadas aos métodos de treinamento, seleção e organização industrial. De modo ainda mais significativo, os psicólogos podem garantir que os projetistas se concentrem primeiramente em fazer os sistemas adequados às pessoas, e não o contrário: supor que o usuário irá se adaptar ao sistema.

Nos últimos anos, os ergonomistas vêm estudando a maneira como as pessoas reagem aos softwares complexos — a chamada "interface do usuário". No início da computação, os usuários eram profissionais bem treinados, altamente motivados, preparados para aceitar desconforto e inconveniências, dispostos a atingir um nível de competência técnica proporcional à potência do equipamento. Hoje o usuário é literalmente um leigo, uma pessoa comum, pouco tolerante com máquinas temperamentais ou exigentes. Se tivesse de aprender uma linguagem de informática para utilizar o caixa eletrônico de um banco, ele provavelmente encarregaria alguma instituição financeira de cuidar de seus negócios. E não são apenas os usuários ocasionais que têm problemas de interface com as máquinas. Os sistemas de bancos de dados que colocam megabytes de informações em cada mesa de escritório são, em toda parte, subutilizados ou utilizados erradamente; extrair deles informações exige fluência em linguagens computacionais complexas. Pesquisadores e usuários esperam que a próxima geração de sistemas aplicativos torne possível a comunicação com o banco de dados ou o modelo financeiro em linguagem próxima à comum. Nesse tipo de sistema, um primeiro nível do terminal traduzirá a entrada do usuário para a linguagem de comando do sistema, pedindo-lhe que especifique melhor suas solicitações e explicando ou expandindo as respostas do sistema.

Há muitos modos de auxiliar o usuário, inclusive recursos de "auxílio em linha", incorporados aos softwares mais avançados. A pesquisa no campo da inteligência artificial levou ao desenvolvimento de programas especialistas, que reúnem grande número de conhecimentos e podem explicar o processo pelo qual se chega a decisões. Acredita-se que essas técnicas ajudarão a criar "módulos de orientação" para auxiliar os usuários. Esse tipo de software "inteligente" dá, entretanto, origem a uma questão filosófica: o que é uma boa explicação e para quem se dirige? O programador poderia querer uma explicação das estruturas dos dados e dos processos de um sistema; já o usuário financeiro preferiria um comentário sobre os meios de se atingir um objetivo comercial. Tais diferenças são problemas de semântica e de lingüística para os ergonomistas e os cientistas de computação.

Outra área nebulosa no estudo da interface do usuário é conhecida pelos psicólogos como "modelagem". As pessoas utilizam modelos mentais (por exemplo, estereótipos nacionais ou raciais) para preencher lacunas em seu conhecimento; muitos de nós emitiríamos, sem titubear, opiniões sobre a aparência, pontos de vista, personalidade e posições políticas de um estranho simplesmente e a partir do conhecimento de seu trabalho — funcionário público, cobrador de ônibus, maestro. De modo semelhante, as pessoas abordam os computadores com idéias estereotipadas sobre o poder e comportamento das máquinas, podendo imaginá-las mais inteligentes e com maior conhecimento do que os seres humanos no desempenho do mesmo tipo de tarefa. Quando encontram respostas impessoais e lacônicas dadas por muitos programas de softwares (especialmente em relação a entradas incorretas), elas podem considerar os computadores mal-educados, inamistosos e até mesmos hostis — julgamentos totalmente impróprios para máquinas. Essa percepção personalizada da máquina afeta toda a interação, muitas vezes levando a respostas inadequadas de ambos os lados do console.

Programadores que tentam incentivar a interação podem agravar o problema, pela introdução de características que fazem o programa parecer mais inteligente do que realmente é. Por exemplo, mensagens amistosas como "Olá, João, eu sou o espírito do banco de dados" podem incentivar o usuário a responder no mesmo estilo, muitas vezes com resultados catastróficos e correspondente desestímulo.

A interação requer uma abordagem mais hábil, envolvendo reflexão e preocupação com as pessoas, bem como admissão de suas reações complexas aos computadores e ao trabalho. Ergonomia, portanto, significa bem mais que inclinar o vídeo ou pintar o computador de lilás.

Condições de trabalho
De modo geral, admite-se que as pessoas no trabalho têm seu desempenho afetado por elementos físicos fáceis de notar, como a altura da mesa de trabalho, a temperatura ambiente e os níveis de ruído. Efeitos mais sutis, entretanto, como cores, luzes e percepção espacial, estão sendo reconhecidos agora como aspectos igualmente importantes no local de trabalho, em especial onde há computadores. Ao instalar um novo sistema, o analista deve considerar todos esses fatores, além de escolher o hardware e o software.

BASE LUNAR

Este programa auto-explicativo permite explorar toda a capacidade gráfica do TK 90X. Ao mesmo tempo, é um fascinante jogo de ação, que se apresenta em dois níveis de dificuldade.

Você tem por incumbência defender uma base lunar, sujeita a freqüentes ataques de naves alienígenas que tentam destruir suas reservas de combustível. Para tanto, dispõe de um canhão laser, cuja mira aparece na tela. O movimento do canhão é comandado pelas teclas [5], [6], [7] e [8]: aponte para o inimigo e dispare com o [0].

A tática usada pelos adversários é a de aproximar-se ao máximo dos tanques de combustível, a fim de não falharem no ataque e de tentar destruí-los completamente. Assim, quanto menor for a distância do inimigo abatido, mais pontos você ganhará. Mas cuidado, pois o menor erro nesse instante crítico pode custar caro. Você pode optar entre dois níveis de jogo: [0], difícil; e [1], fácil.

O algoritmo do programa foi construído utilizando lógica modular; assim, por meio do GOSUB, o programa principal gerencia várias sub-rotinas, o que possibilita a observação de trechos isolados do mesmo.

Um efeito que pode ser indesejável é a lentidão do programa. Contornamos isso facilmente eliminando alguns comandos de som e sub-rotinas ou alterando o algoritmo para lógica estruturada. Outro recurso consiste no emprego de linguagem de máquina.

O programa que segue já inclui uma rotina geradora de caracteres nas linhas 750, 760, 780, 786 e 790. Se você optar por redefinir os caracteres, basta omitir essas linhas ao digitar e seguir as instruções de redefinição. Do contrário, deve mudar os caracteres criados pelos modos gráficos das letras A, B, C, D, E, F, e G da seguinte forma:

Posição	Caracteres usados
Afastada	B
Média	C, D
Próxima	C, E, D
Mira	A
Explosão	F, G

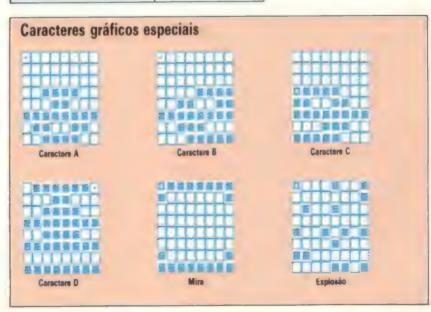
Redefinição de caracteres

Os caracteres gráficos especiais precisam ser definidos antes da digitação do programa. O TK 90X dispõe de um editor de caracteres, acessado pela função UDG 2, o que você deve fazer logo depois de ligar o computador.

Quando se acessa a função, surgem na tela as letras do alfabeto, de A até U, representadas numa matriz 8 x 8. Para criar os caracteres, digite a letra a ser desenhada; pressione em seguida as teclas [Caps Shift] ou [Simbol Shift] e a [1], simultaneamente, para limpar a matriz. Esse procedimento faz aparecer um ponto no canto superior esquerdo, que pode ser deslocado em qualquer direção no interior da matriz, por meio das teclas [5], [6], [7] e [8]. É com esse ponto que se formam os caracteres. Para tanto, escureça os quadros em que ele aparece, com uma das teclas de movimento pressionada junto com a [Caps Shift].

Apague o ponto também com uma das teclas de movimento mais a [Simbol Shift], ambas pressionadas ao mesmo tempo.

Para criar o caractere seguinte, digite a tecla correspondente, mas sem pressionar [Caps Shift]. Esta, acionada simultaneamente com [0], permite sair do modo de edição de caracteres; e, com a [9], possibilita acessar os caracteres gerados e obter o cursor G. Digitando a tecla correspondente, você obtém o símbolo gráfico criado no modo UDG 2.

EFEITO DE APROXIMAÇÃO	
Posição	**Caracteres usados**
Afastada	A
Média	B, C
Próxima	B, D, C

```
  1 REM BASE LUNAR
 20 GOSUB 900
 30 GOSUB 700
 35 GOSUB 800
 40 GOSUB 300
 45 GOSUB 100
 50 IF ht<>1 THEN GOSUB 100
 70 PRINT INK 5; OVER 0;AT 19,6
;po;" ";AT 19,25;sc;AT 20,6;lz;A
T 20,15;tno;AT 20,25;ki
 80 IF po<0 THEN GOTO 600
 90 GOTO 40
100 LET uxo=ux: LET uyo=u
101 LET u$=INKEY$
105 IF u$="" THEN RETURN
106 IF u$="0" OR u$="1" THEN GO
TO 150
107 IF u$<"5" OR u$>"8" THEN RE
TURN
110 LET ux=ux+(u$="8" AND ux<29
)-(u$="5" AND ux>2)
120 LET uy=uy+(u$="6" AND uy<13
)-(u$="7" AND uy<13)-(u$="7" AND
 uy>2)
125 IF ux=uxo AND uy=uyo THEN R
ETURN
130 PRINT INK 7;AT uyo,uxo;"□";
 INK 6;AT uy,ux;"□": LET po=po-1
  SOUND .005,20-uy: RETURN
150 INK 5: LET po=po-10
155 LET lz=lz+1: LET lcx=ux*8+4
 LET lcy=(21-uy)*8+4
160 FOR n=1 TO 2: SOUND .04,6:
SOUND .02,12: PLOT 8,56: DRAW lc
x-8,lcy-56: PLOT 247,56: DRAW lc
x-247,lcy-56: INK 7: NEXT n
175 INK 6: FOR n=1 TO 2: PRINT
AT uy,ux-1;">▒<": SOUND .03,24:
INK 7: NEXT n
180 IF NOT (uy=ty AND ux>=tx-df
 AND ux<tx+df+LEN t$) THEN INK 5
: RETURN
200 OVER 1: LET ht=0: LET ki=ki
+1: LET sc=sc+100-LEN t$*10
210 FOR n=1 TO 4: PRINT AT uy-1
,ux-1;"\*/";AT uy,ux-1;"▒▒▒";AT
uy+1,uy-1;"/*\": SOUND .1,36: SO
UND .05,6: INK 7: NEXT n: LET ht
=1
220 RETURN
300 IF ht=1 THEN GOTO 360
302 LET tyo=ty: LET txo=tx
305 LET u$=t$: LET tx=tx+1
310 IF ty>=12 THEN LET tc=-1
311 IF ty<=4 THEN LET tc=1
312 LET ty=ty+tc+(INT (RND*2) A
ND ty<14)-(INT (RND*2) AND ty>3)
315 LET t$="": LET t$=("♣" AND
tx<9)+("▬▬" AND tx>=9 AND tx<18)
+("▬▬▬" AND tx>=18)
330 PRINT INK 4;AT tyo,txo;u$;A
T ty,tx;t$: SOUND .02,-12
343 IF ty>8 AND RND<.7 AND ((tx
>16 AND tx<21) OR tx=24) THEN IN
K 2: FOR m=1 TO 2: FOR n=-1 TO 2
: PLOT tx*8+12,175-(ty*8)-12: DR
AW n*3,-40: SOUND .003,30: NEXT
n: INK 7: PRINT INK 6;AT 21,0;"▒
▒▒▒▒▒▒▒▒▒▒▒▒▒▒": LET po=po-9: N
EXT m
355 IF tx<=27 THEN RETURN
360 PRINT INK 7;AT ty,tx;t$: LE
T tno=tno+1: LET tx=2: LET ty=7+
INT (RND*6): LET tc=1: LET ht=0
375 LET t$="♣"
380 PRINT INK 4;AT ty,tx;t$
385 RETURN
600 FOR n=0 TO 74: INK n/10: SO
UND .36,m-50: PRINT AT 10,8;"- F
IM DA MISSAO -": NEXT n
640 FOR n=0 TO 74: BORDER 7-n/1
0: SOUND .05,n-20: NEXT n
660 INPUT INK 6;TAB 6;"OUTRA MI
SSAO ? ";u$
685 IF u$="n" THEN GOTO 9999
690 RESTORE : GOTO 30
700 INPUT INK 2;"Dificuldade?:
0(dificil) ou 1 ";r$: IF r$<>"0"
 THEN LET r$="1"
705 LET df=VAL r$
710 LET ux=10: LET uy=10
715 LET uxo=ux: LET uyo=uy
720 LET tx=4: LET ty=2: LET txo
=tx: LET tyo=ty: LET tc=1
730 LET po=999: LET sc=0: LET k
=0: LET lz=0: LET ht=0
740 LET t$="♣": LET tno=1
750 FOR n=USR "a" TO USR "g"+7
760 READ d: POKE n,d: NEXT n
770 RETURN
780 DATA 255,129,0,0,0,129,129,
255,0,0,0,24,[illegible],255,56,0
786 DATA 0,0,2,7,9,255,9,2,0,0,
32,224,144,255,144,64,112,32,210
,255,126,255,126,189
790 DATA 16,68,16,8,230,0,20,16
,149,88,40,231,82,20,74,145
800 BORDER 0: PAPER 0: INK 7
805 OVER 0: CLS
810 PLOT 3,26: DRAW 247,0: DRAW
0,144: DRAW -247,0: DRAW 0,-144
811 LET gy=34: LET gc=6
812 FOR n=1 TO 6: PLOT 5,gy
813 DRAW 243,0: LET gc=gc-1: LE
T gy=gy+gc: NEXT n
815 FOR n=1 TO 70: INK 2+RND*6:
 PLOT 10+RND*230,70+RND*90: DRAW
 RND,0: NEXT n: INK 7
820 LET gc=1: LET gy=54
821 FOR n=5 TO 247: LET gy=gy+g
c+INT (RND*3)-1: PLOT n,gy: DRAW
 0,-(RND*(gy-55) AND gy>54)
822 IF RND<.1 THEN LET gc=-gc
823 IF gy>61 THEN LET gc=-INT (
RND*2.5)
824 IF gy<54 THEN LET gc=INT (R
ND*2.5)
826 NEXT n
828 PRINT INK 5;AT 15,1;"▙";AT
15,30;"▟"
830 FOR m=0 TO 60 STEP 20
832 IF m=40 THEN NEXT m
834 FOR n=38 TO 51: PLOT 140+m,
n: DRAW 20,0,.7: NEXT n: DRAW -2
0,0,.7: NEXT m
850 FOR n=1 TO 5: CIRCLE INK 4;
23,140,n: NEXT n: OVER 1
870 PRINT INK 6;AT uy,ux;"□"
880 PRINT AT ty,tx;"♣"
885 PRINT AT 19,0;"forca";AT 19
,19;"score";AT 20,0;"laser ";AT
20,12;"no";AT 20,19;"KILLS"
890 SOUND 1,9: RETURN
900 BORDER 7: PAPER 7: INK 0
905 OVER 0: CLS
910 PRINT AT 0,10;"BASE LUNAR"
915 PRINT "VOCE CONTROLA UMA ES
TACAO DE LASER QUE PROTEGE A LUA
 DOS INIMIGOS"
930 PRINT "HA UM LIMITE DE FORC
A DE DEFESA. ATINGIDO ESSE LIMIT
E , TERMINA SUA AVENTURA"
940 PRINT "PARA ATIRAR TECLE <0
> OU <1>"
950 PRINT "TECLE <5> OU <8> PAR
A MOVIMENTAR LATERALMENTE A MIRA
 DO LASER E <6> OU <7> PARA MOVI
MENTOS VERTICAIS"
985 RETURN
9999 BRIGHT 0: FLASH 0: OVER 0
INK 0: PAPER 7: BORDER 7
```

SUB-ROTINAS EMPREGADAS

Linhas	Sub-rotinas
900-990	Instruções do jogo
700-790	Determinação das variáveis e definição de caracteres especiais
800-890	Desenho da tela
300-340	Movimento dos inimigos
100-125	Movimento da mira do canhão

Programa, de autoria de Fábio Polônio, cedido pela revista **Microhobby**, edição n.º 21.

SALTOS E DESVIOS

A linguagem ASSEMBLY utiliza flags e instruções de salto relativo para direcionar o fluxo de controle do programa. Veremos aqui como esses recursos são combinados para criar loops e tabelas de dados.

Para que possamos utilizar os vários modos de endereçamento da CPU, especialmente os endereços indexados, é necessário em primeiro lugar construir loops. Sem essa estrutura básica, ficaríamos na situação de um programador em BASIC que sabe armar matrizes, mas não conhece o comando FOR-NEXT. Não há na linguagem ASSEMBLY uma estrutura automática de loop como o FOR-NEXT (embora o Z80 possua uma instrução bem semelhante), mas podemos criar loops do tipo IF-THEN-GOTO. Estes requerem instruções que tomam decisões ou expressam condições, e irão modificar a ordem de execução das instruções do programa.

A tomada de decisões na linguagem ASSEMBLY baseia-se nos flags do registrador de estado do processador. Eles mostram o efeito da última condição executada sobre o acumulador; por isso são às vezes chamados de flags de condição. Todos atuam na tomada de decisões, mas consideraremos apenas dois deles — o flag zero (Z) e o de transporte (C, "carry").

O estado desses flags determina se o processador irá executar a próxima instrução do programa ou se irá saltar para alguma outra em qualquer ponto do programa. Para que o processador decida se irá continuar na seqüência ou saltar para outra instrução, ele deve aceitar (ou então modificar) o endereço existente no contador do programa. Esse registrador sempre contém o endereço do próximo comando em código de máquina a ser obedecido. Ao começar a executar uma instrução, o processador carrega o código de operação dessa ordem. O endereço contido no registrador é incrementado de acordo com o número de bytes da instrução, de modo que o contador do programa passa a indicar o código de operação seguinte. Se a instrução atual fizer o contador do programa indicar um endereço em outro ponto, então realiza-se um salto, ou desvio (abreviado por J, de "jump", ou B, de "branch").

No processador 6502, o comando BEQ altera o contador do programa se o flag zero está ligado; e BCS fará o mesmo se o flag de transporte estiver ativado. No Z80, esses comandos são JR Z e JR C, respectivamente. Esses quatro códigos de operação chamam-se instrução de desvio, pois representam um ponto de desvio no fluxo de controle do programa. Seu operando é um número de um só byte, que, acrescentado ao endereço do contador, produz um novo endereço. Veja o que acontece na execução desse programa.

ORG $5E00			
6502		Z80	
5E00	ADC #S34	ADC	A,S34
5E02	BEQ S03	JR	Z,S03
5E04	STA S5E20	LD	(S5E20),A
5E07	RTS	RET	

Se a instrução ADC em $5E00 produzir um resultado zero no acumulador (improvável, mas pode acontecer), as instruções BEQ ou JR Z em $5E02 acrescentarão $03 ao conteúdo do contador do programa. Portanto, a próxima ordem a ser executada será o retorno a $5E07, saltando a instrução em $5E04.

À primeira vista, esse salto parece incorreto. Poderíamos pensar que, se a instrução em $5E02 acrescenta $03 ao contador do programa, o endereço nele armazenado deverá tornar-se $5E05. Mas é importante lembrar que o contador do programa sempre indica a próxima instrução a ser obedecida, e não aquela em execução no momento. Dessa forma, no início da execução da instrução em $5E02, o contador do programa conterá o endereço $5E04, que é a posição do comando seguinte. Quando se acrescenta $03 a $5E04, o resultado será $5E07, o endereço da próxima instrução.

É importante assinalar que o processador não tem a capacidade de checar se os endereços indicados são corretos. Se por engano determinarmos que o deslocamento será de $02, o contador do programa, contendo zero, será incrementado de $5E06 como o endereço do código de operação da instrução seguinte. Em nosso programa correto, $5E06 contém o valor $5E, que é o byte mais alto do operando da instrução em $5E04. Contudo, o processador não pode avaliar se esse valor é ou não o comando correto. Ele considera $5E como um código de operação válido, e passará a executá-lo, supondo os bytes após $5E06 como os operandos da instrução. O resultado será a falha total do programa. Esse tipo de deslocamento mal calculado constitui um dos erros mais freqüentes na programação em código de máquina.

Ao programar em linguagem ASSEMBLY, porém, o cálculo dos deslocamentos deixa de ser um problema porque quem o executa é o programa montador, ou compilador. Assim, em vez de determinar um deslocamento hexadecimal co-

mo operando da instrução de desvio, fornecemos apenas um endereço simbólico da instrução a ser executada. Isso facilita muito a compreensão dos programas em linguagem ASSEMBLY. O programa assemblador transforma o endereço simbólico num endereço absoluto, calcula o deslocamento necessário para atingir tal endereço e insere esse deslocamento na instrução em código de máquina. O endereço simbólico chama-se rótulo e equivale a um número de linha nos programas em BASIC.

Examinemos mais detalhadamente o uso dos rótulos. Um rótulo é um string alfanumérico colocado no início de uma instrução em linguagem ASSEMBLY. O compilador o considera como um símbolo de 2 bytes, representando o endereço do primeiro byte da instrução. Assim, podemos reescrever nosso programa utilizando o rótulo EXIT para representar o ponto de saída:

ORG $5E00				
	6502		Z80	
5E00	ADC	#$34	ADC	A,$34
5E02	BEQ	EXIT	JR	Z,EXIT
5E04	STA	$5E20	LD	($5E20),A
5E07 EXIT	RTS		RET	

A instrução em $5E02 significa agora "IF o valor do acumulador for zero THEN GOTO o endereço representado por EXIT". Essa versão do programa fica mais compreensível que a anterior, diminuindo a possibilidade de cálculo errado quanto ao ponto de destino do salto.

Podemos agora criar um loop com as instruções de desvio e os rótulos EXIT e START, indicando este último o ponto inicial:

ORG $5E00				
	6502		Z80	
5E00 START	ADC	#$34	ADC	A,$34
5E02	BNE	START	JR	NZ,START
5E04	STA	$5E20	LD	($5E20),A
5E07 EXIT	RTS		RET	

Observe a utilização das novas instruções de desvio BNE e JR NZ, ambas significando: "Desviar se o acumulador não for igual a zero". Vejamos o efeito desse programa. A primeira instrução acrescenta $34 ao acumulador. Se o resultado não for igual a zero, o programa desviará, retornando a $5E00, que é o endereço representado por START. Mais uma vez, acrescenta-se $34 ao acumulador e, conforme o resultado, ocorrerá novo desvio. Esse loop continuará repetindo a instrução ADC, até que se satisfaça a condição de desvio. Quando o conteúdo do acumulador for igual a zero, não ocorrerá o desvio em $5E02; em vez disso, será executada a instrução em $5E04.

Isso equivale a um loop IF-THEN-GOTO em BASIC; contudo, fica difícil compreender como o acumulador pode chegar a zero, pois ele é incrementado em $34 a cada execução do loop. Como poderá totalizar zero? O acumulador é um registrador de apenas um byte. Se a adição resulta num número de 2 bytes, ativa-se o flag de transporte do registrador de estado do processador; assim, o acumulador conterá o byte mais baixo do resultado. Se o acumulador contiver $CC, por exemplo, o acréscimo de $34 resultará em $0100, um número de 2 bytes. O flag de transporte será ativado, e o acumulador conterá o byte mais baixo desse resultado — $00. Assim, o conteúdo do acumulador se tornará zero e, como resultado, o flag zero será ativado.

Tendo em mente esse resultado, poderíamos reescrever o programa usando outra condição de desvio, incorporando o estado do flag de transporte, e não do flag zero:

ORG $5E00				
	6502		Z80	
5E00 START	ADC	#$34	ADC	A,$34
5E02	BCC	START	JR	NC,START
5E04	STA	$5E20	LD	($5E20),A
5E07 EXIT	RTS		RET	

Nessa versão, a instrução em $5E02 significa: "Se o flag de transporte estiver desligado, desviar para START". Quando o acréscimo de $34 do acumulador resultar num valor maior que $FF, o flag de transporte será ativado, e não haverá retorno ao endereço START.

Contadores de loop

O desvio de acordo com a condição dos flags de transporte e zero parece ser um recurso bastante limitado; permite, contudo, inúmeras possibilidades de tomadas de decisão. O que nos falta agora é um contador de loop, para contar, por exemplo, o número de execuções de um loop ou para determinar a saída dele após um certo número de iterações. Atinge-se o primeiro objetivo facilmente, pelo emprego de um registrador de índice da CPU para manter o contador e de uma instrução de incremento para atualizá-lo:

6502		
0000	ORG	$5DFD
5DFD	LDX	#$00
5DFF START	INX	
5E00	ADC	#$34
5E02	BCC	START
5E04	STX	$5E20
5E07 EXIT	RTS	

Z80		
0000	ORG	$5DFA
5DFA	LD	IX,$0000
5DFE START	INC	IX
5E00	ADC	A,$34
5E02	JR	NC,START
5E04	LD	($5E20),IX
5E08 EXIT	RET	

A nova estrutura causou várias alterações no programa. Em primeiro lugar, as instruções inseridas no início exigem um novo endereço ORG. Essas instruções têm efeito semelhante nos processadores 6502 e Z80, mas seus tamanhos são diferentes, de modo que os endereços de posição diferem nas duas versões do programa.

Em segundo lugar, usamos novas versões das instruções de carga (LDX, LD IX) e armazenamen-

to (STX, LD(),IX) para colocar um valor inicial de $00 no registrador de índice da CPU. O registrador X do 6502 tem apenas 1 byte, mas o registrador IX do Z80 tem 2. Os registradores de índice têm funções especiais, mas são, em princípio, memória dinâmica da CPU, tal como o acumulador; aqui os utilizamos como acumuladores adicionais, para a contagem do loop. Na versão do 6502, quando ocorre a saída do loop, o conteúdo do registrador X é armazenado em $5E20. Na versão do Z80, o byte mais baixo do registrador IX é armazenado em $5E20, e o byte mais alto em $5E21.

Em terceiro lugar, uma nova instrução substitui a ADC como ponto inicial do loop: INX e INC IX são comandos de incremento, que acrescentam $01 ao conteúdo do registrador de índice. Isso atualiza o valor do contador de loops a cada iteração.

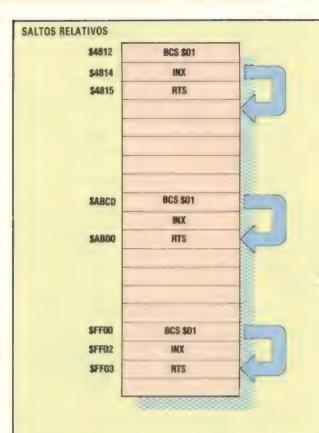
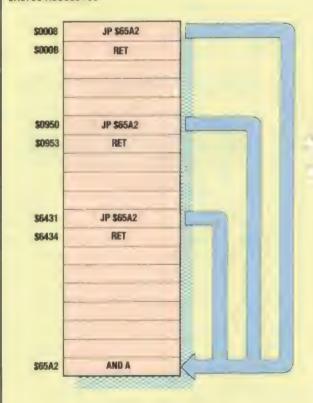
Podemos interpretar o programa assim: "Zerar o contador de loops, iniciar o loop incrementando o contador, acrescentar $34 ao acumulador e voltar para o início do loop se o flag de transporte estiver desligado. Do contrário, armazenar em $5E20 o conteúdo do contador de loops".

Uma modificação complementar aumentará muito a utilidade e abrangência do programa:

6502		
0000	ORG	S5DFA
5DFA	LDX	#S00
5DFC START	STA	S5E22,X
5DFF	INX	
5E00	ADC	#S34
5E02	BCC	START
5E04	STX	S5E20
5E07 EXIT	RTS	

Z80		
0000	ORG	S5DF7
5DF7	LD	IX,S5E00
5DFB START	LD	(IX+S22),A
5DFE	INC	IX
5E00	ADC	A,S34
5E02	JR	NC,START
5E04	LD	(S5E20),IX
5E08 EXIT	RET	

Ambas as versões do programa têm o mesmo efeito: criam, na posição $5E22, uma tabela de armazenamento dos valores sucessivos do acumulador, à medida que transcorre o programa, e armazenam em $5E20 o valor final do contador de loops, que equivale ao número de bytes da tabela.

A versão 6502 chega a esse resultado por meio da instrução STA $5E22,X, que significa "Acrescentar o conteúdo do registrador X ao endereço-base $5E22; então armazenar o conteúdo do acumulador do endereço assim formado". A instrução STA está aqui no modo indexado direto absoluto, isto é, ela usa o registrador X como um índice para modificar o endereço-base $5E22. Como o registrador X é inicializado em $00 e subseqüentemente incrementado a cada iteração, o valor inicial do acumulador será armazenado

Saltos relativos
A maioria das instruções de desvio, como BCS (que significa "Desviar se o flag de transporte estiver ligado") e JR NZ ("Desviar se o acumulador for não zero"), age de acordo com a condição do registrador de estado do processador, e utiliza o modo de salto relativo para redirecionar o fluxo de controle do programa.
A alternativa é o salto absoluto.

No exemplo, a instrução BCS $01 sempre causa um salto relativo de 1 byte para a frente (quando ocorre o salto, pois ela está condicionada ao estado do flag de transporte). Nesse salto, não importa em qual endereço se encontra o código de máquina. Aqui, a instrução BCS $01 é sempre seguida pela INX, que possui um só byte; portanto, quando se ativa o flag de transporte, BCS faz o programa saltar a instrução INX.

Saltos absolutos
Neste exemplo, a instrução JP $65A2 ocasiona um salto incondicional toda vez que aparece. Seu efeito é redirecionar a execução do programa para o endereço que forma seu operando — nesse caso, $65A2. Não se realiza qualquer teste, e o endereço da instrução em execução não é significativo, pois a execução do programa sempre continua a partir do endereço especificado.

Ambos os modos de salto apresentam vantagens e desvantagens, mas o critério mais importante na escolha entre um salto relativo e um absoluto está no reposicionamento. É bastante comum, na programação em ASSEMBLY, escrever-se uma rotina, montá-la num dado endereço ORG, e depois reutilizá-la com a mesma forma, mas com um valor ORG diferente. Se todos os saltos na rotina forem relativos, a modificação dos endereços das instruções não terá qualquer importância, e o programa fluirá sem dificuldade, seguindo as rotas desejadas; entretanto, se algum dos saltos for absoluto, quando se montar a rotina com outro ORG, os saltos continuarão a direcionar o controle para o endereço especificado, que pode então não ter qualquer significado para a rotina.
Os saltos relativos são reposicionáveis, mas os absolutos não.

em $5E22, o próximo valor em $5E23, e assim sucessivamente. Após a saída do loop, STX armazenará na posição $5E20 o valor final do contador de loops.

A versão do Z80 utiliza o registrador IX para indicar o endereço atual de armazenamento, empregando o byte mais baixo de IX como contador de loops. A instrução LD IX,$5E00 coloca no registrador IX o endereço-base $5E00; assim, o byte mais baixo de IX conterá $00. A instrução LD (IX + $22),A, de aparência um pouco estranha, significa "Acrescentar o endereço contido em IX e $22 e armazenar o conteúdo do acumulador no endereço assim formado". Como IX é inicializado em $5E00 e incrementado a cada iteração do loop, o valor inicial do acumulador será armazenado em $5E22, o seguinte em $5E23, e assim por diante. O byte mais baixo de IX vai registrando o número de iterações do loop, valor que é armazenado em $5E20, quando o loop se encerra. A instrução LD(IX + $22),A encontra-se aqui no modo de endereçamento indexado indireto absoluto, de complexidade um pouco maior que na versão do 6502, porém muito mais potente.

Agora já examinamos como a linguagem ASSEMBLY manipula loops e tabelas, técnicas de programação de grande utilidade. No próximo artigo veremos o uso prático de ambas.

Exercícios

Vimos neste artigo muitos pontos importantes — e talvez complexos —, e somente a prática das novas instruções e modos de endereçamento permitirá compreendê-los bem.

Use o programa compilador para montar e gravar as rotinas apresentadas neste artigo. Ao executar uma rotina, use o modo de depuração (debug) para examinar as posições de memória afetadas. É uma boa idéia inicializar sempre essas posições com uma constante identificável — $FF, por exemplo — antes da execução, de modo que posteriormente seja possível verificar se a memória foi alterada pelo programa. Para isso, use o comando debug Alter, ou mesmo o comando debug Move. Como sempre, lembre-se de que os endereços dados no programa são apenas exemplos; escolha, pois, endereços adequados à sua máquina.

Flag — levantando a bandeira
A palavra flag, que em inglês significa bandeira, indica a ocorrência de uma condição num programa, por analogia às bandeirinhas usadas em sinalização. Um flag nada mais é que um bit utilizado como sinal. No estado zero, ou desligado, representa uma bandeira abaixada; quando, porém, ocorre determinada condição na execução do programa, o flag passa a valer 1, ou seja, a bandeirinha levanta. Desvia-se, assim, o fluxo de comando do programa de acordo com o estado do flag.

As instruções de desvio condicional, como vimos, dependem do conteúdo do PSR (Processador Status Register, "registrador de estado do processador"). Uma razão para se utilizar a opção de apresentação binária é possibilitar a verificação do conteúdo do PSR antes e depois da execução de uma instrução, e então observar as alterações nos flags. Não há, na linguagem ASSEMBLY uma única instrução para armazenar o conteúdo do PSR; assim, temos de usar os seguintes comandos:

Z80	
3E00 F5	PUSH AF
3E01 F5	PUSH AF
3E02 E1	POP HL
3E03 22 lo hi	LD (STORE1),HL
3E06 F1	POP AF

6502	
3E00 48	PHA
3E01 08	PHP
3E02 48	PHA
3E03 08	PHP
3E04 68	PLA
3E05 8D lo hi	STA STORE1
3E08 68	PLA
3E09 8D lo' hi'	STA (1+STORE1)
3E0C 28	PLP
3E0D 68	PLA

Essa seqüência de instruções armazenará o conteúdo atual do PSR no byte endereçado por STORE1 (que representa um endereço apropriado para sua máquina), enquanto o conteúdo do acumulador será armazenado em (1 + STORE1). Para usar essas instruções, basta inseri-las como um bloco, antes e depois da instrução cujo efeito você deseja observar. Entretanto, lembre-se de acrescentar 2 ao valor de STORE1 toda vez que inserir esse bloco. Após executar o programa, verifique a parte da memória onde se armazenaram os conteúdos do PSR e do acumulador.

Talvez lhe ocorra tratar esse bloco como uma subrotina, em vez de entrar com ele toda vez que necessário. Há no ASSEMBLY um equivalente do comando GOSUB do BASIC, mas seu uso aqui complicaria a programação, pois emprega a técnica do empilhamento. Ele iria interferir no uso que o bloco faz da mesma lista (PLA, PUSH, PHP etc. são manipulações de pilhas, que posteriormente explicaremos). Observe a diferença de tamanho entre a rotina do Z80 e a do 6502: isso se deve aos registradores de 2 bytes do Z80 e às instruções associadas.

SID

A SID Informática S.A. responde pela produção nacional — com tecnologia própria — de sistemas de automação bancária e comercial, de micro e minicomputadores.

Fundada em 1978, a SID Informática, empresa do grupo Sharp, é especializada na fabricação de micro e minicomputadores, bem como de sistemas de automação bancária e comercial, com 90% de tecnologia nacional.

Mas a história desse conglomerado de 43 empresas começa bem antes. Em 1961 foi fundado o conjunto de empresas Machline, que, em 1969, obteve a concessão para comercializar com exclusividade no Brasil os produtos da empresa japonesa Sharp Corporation. A partir daí, criou-se a Sharp Equipamentos Eletrônicos, que, apesar de possuir o mesmo nome da empresa japonesa, é totalmente nacional.

Quando instalou sua primeira unidade industrial na Zona Franca de Manaus, em 1972, a Sharp produzia calculadoras eletrônicas. Em 1974 foi uma das primeiras empresas a fabricar televisores em cores e, no ano seguinte, equipamentos de áudio. Hoje a Sharp Equipamentos Eletrônicos responde também por videocassetes e videocâmaras, empregando mais de 2.000 funcionários em suas quatro unidades industriais. Para aumentar a nacionalização de seus produtos, a empresa passou a investir na produção de componentes eletrônicos.

Na informática

Também nesse setor, a Sharp abarca parte considerável do mercado, através dos Sistemas de Informação Distribuída ou, simplesmente, SID Informática. Esta produz terminais financeiros, micro e minicomputadores e diversos outros equipamentos do setor de informática.

Para a automação bancária, a SID oferece uma linha de equipamentos que permite o serviço computadorizado de uma agência ou a integração de várias agências controladas por um computador central de grande porte. Esse núcleo de controle — o SID 3500 — processa todos os dados da agência e concentra as informações dos terminais de caixa, de retaguarda, de consulta e administrativos.

Na área comercial, a SID possui o terminal PDV (ponto de venda), que funciona como caixa registradora eletrônica e microcomputador.

O PDV SID-6000 tem processador de 16 bits e permite, entre outros serviços, a leitura de preços por código de barras, integração em redes de processamento para cadeia de lojas, leitura de cartão magnético para transferência automática de fundos, controle da contabilidade do estabelecimento e análise do giro do estoque.

Na área de microcomputadores, a SID comercializa três modelos: o 3000, o 3800 e o 3900. Todos equipados com microprocessador Intel 8085, memória MOS dinâmica de 64 Kbytes e ROM de 16 Kbytes. Possuem sistema operacional compatível com CP/M e interfaces que permitem ligação com outros micros, mínis ou computadores de grande porte, inclusive emulando terminais de outros fabricantes.

Os minicomputadores da série 5800 também apresentam três modelos, com capacidades variando entre 126 Kbytes e 1 Mbyte. Possuem processador central multiprogramado, e usam palavras de 16 bits.

Os mínis da SID destacam-se por utilizar o avançado banco de dados Adabas. Em 1985, a Sharp lançou o micro Hot-Bit, dos primeiros no Brasil a adotar o padrão MSX.

Em 1984, a SID assumiu o controle da fábrica de semicondutores da Philco, que passou a se chamar SID Microeletrônica. É a única empresa da América Latina que executa todas as fases de produção de semicondutores, componente fundamental para fabricação de equipamentos eletrônicos.

ENCICLOPÉDIA ELETRÔNICA

Uma das maiores vantagens do uso dos computadores na educação é o acesso a bancos de dados grandes e sempre atualizados. Examinaremos alguns programas especificamente elaborados para os jovens alunos.

Há muitos benefícios educacionais em se obterem informações de uma "biblioteca eletrônica"; por exemplo, acesso imediato, sem que seja preciso sair da sala, nem mesmo da carteira, para conseguir os dados. Isso elimina o problema de alunos que "não têm o livro". Além disso, registros computadorizados são facilmente atualizáveis, enquanto as informações em livros de texto não — só poderão ser alteradas numa eventual reimpressão. Os custos da produção de livros representam sério problema ao fornecimento de material didático, e as escolas com freqüência se vêem forçadas a adotar publicações desatualizadas. O pior é que o alto custo do texto impresso também impede a expansão de novos métodos educacionais. Por outro lado, uma escola equipada com computadores requer, para a alteração dos currículos, apenas a troca dos discos mestres.

Os benefícios educacionais evidenciam-se particularmente na combinação de microcomputadores e recursos de telecomunicação para serviços de bancos de dados on line (conexão direta). Ter, por exemplo, toda uma enciclopédia em discos constituiria trunfo extremamente útil. Se pudermos encontrar a informação desejada pesquisando nos índices alfabéticos comuns, será perfeitamente adequado pegar um exemplar do livro da estante. Mas, se por exemplo quisermos conhecer todos os marsupiais que vivem

O mundo que as rodeia
A despeito do receio de que o uso de computadores nas escolas confinaria as crianças ao estreito mundo das telas de vídeo, a prática tem mostrado que o software de banco de dados, em particular, pode conduzir a todo tipo de atividades "extracomputador". A necessidade de coletar informações para alimentar o banco de dados pode ser abordada de muitas maneiras diferentes, inclusive a do "passeio pela natureza", onde os alunos são encorajados a observar, registrar e classificar os elementos do meio ambiente.

em tocas, ou listar todos os países que aplicam determinada porcentagem do PNB na educação, vai ser muito cansativo pesquisar e compilar tudo à mão.

Entretanto, uma enciclopédia com, digamos, 26 volumes exigiria, para seu armazenamento, face aos recursos de pesquisa sistemática, uma capacidade imensa de processamento; é aí que os serviços on line entram em ação. Pode-se acessar esse sistema com um modem padrão de 300 ou 1.200 bauds e uma linha telefônica, ficando disponível uma gama vastíssima de informações.

No Brasil, o Serpro e o IBGE colocam seus bancos de dados à disposição do usuário através da Embratel. O Dialog, nos Estados Unidos, é o maior serviço de banco de dados on line do mundo e um dos mais utilizados no campo da educação. Contém duzentos bancos, com mais de cem publicações e 100 milhões de itens de informação — compêndios de todos os assuntos, de história da arte a zoologia. O Dialog fornece, além do autor e título dos documentos ou trabalhos relevantes, também um resumo com uma descrição substancial do artigo — não raro contendo toda a informação de que se necessita. Esta pode então ser impressa ou solicitada como listagem, a um custo bastante razoável. Os bancos de dados disponíveis estão sendo complementados por uma crescente quantidade de software de pesquisa, para ajudar os usuários a extrair e classificar as informações neles contidas.

O banco de dados da firma inglesa Prestel apresenta um tipo diferente de abordagem — o Schools Link ("ligação escolar") —, oferecendo às escolas informações sobre todos os aspectos da computação aplicada à educação, além de proporcionar contatos e debates. Destina-se a professores e contém resenhas de software, de robôs educacionais, de livros e sugestões para projetos; também permite que o equipamento do professor ou da escola acesse programas educacionais.

Entretanto, além da "biblioteca eletrônica", os bancos de dados estão se tornando populares nas salas de aula, aí desempenhando papel ativo. Já existem vários programas que proporcionam aos alunos a experiência de manipular informações e estruturá-las de acordo com suas próprias necessidades.

O banco de dados Factfile

Numa conferência educacional, na universidade de Cambridge, em 1981, discutiu-se a possibilidade de uso dos bancos de dados. Esse debate inspirou a criação de um programa chamado Factfile ("Arquivo de Fatos"), que permite a crianças, mesmo pequenas, montar arquivos de dados sobre qualquer tópico que escolham.

O papel de um banco de dados como o Factfile no estímulo à descoberta e aprendizado ativo logo se torna evidente. Suponhamos que as crianças queiram criar um banco de dados contendo informações sobre seus professores. Primeiramente, terão de decidir as categorias nas

Dinossauros tabulados

A memória permanece
O Factfile proporciona excelente introdução ao gerenciamento de dados. Inclui um arquivo de demonstração chamado Dino, contendo informações sobre dinossauros (alimentação, habitat etc.). As crianças são estimuladas a familiarizar-se com o Dino e a catalogar o conteúdo do arquivo (como acima), antes de passarem à criação de seus próprios bancos de dados.

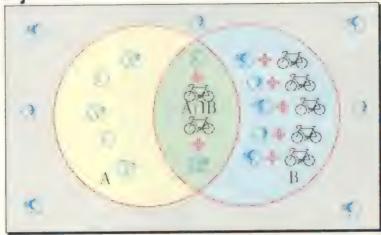
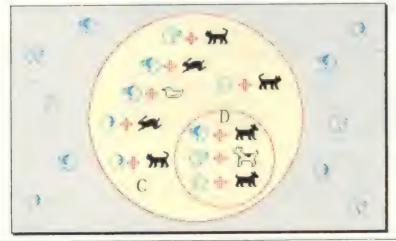
Pedido de pesquisa

Menu do Dino

quais organizarão as informações. Isso dependerá do que já sabem e do que conseguem descobrir sozinhas. Podem, por exemplo, incluir sexo, idade, altura, disciplina lecionada e "tipo de professor". Determinada a estrutura do banco, deverão coletar informações para cada categoria. Nesse ponto, o exercício se transforma numa operação educacional diversificada — no mínimo, as crianças terão de coletar informações físicas. Se forem necessárias medições, terão de determinar as unidades de medida e os dados a incluir. O processo vai se tornando mais complexo à medida que as crianças começam a entender o significado das categorias criadas por elas. O campo "tipo de professor" requer bastante discussão. O que significa: "Que disciplina leciona?" ou "É um professor legal ou chato?".

Encerrada a compilação do banco de dados, outras crianças poderão dirigir-lhe perguntas, correlacionando informações de várias categorias. Talvez queiram listar todas as professoras classificadas na categoria "legal" ou todos os professores de matemática do sexo masculino. E o processo de aprendizagem avança mais um degrau quando as crianças começam a analisar os resultados. O Factfile permite que as crianças gravem seu banco de dados em disco ou cassete e acrescentem mais informações posteriormente, podendo retornar aos arquivos de dados sempre que necessário.

Uma das críticas ao uso de computadores por crianças é que estas se sentam frente a um vídeo, executando atividades sem relevância para o mundo em que vivem. Mas, além de dar às crianças uma experiência direta com bancos de dados, o Factfile leva a inúmeras atividades não computacionais. Ao criar um arquivo com informações sobre a flora local, as crianças teriam de registrar cuidadosamente os detalhes sobre ca-

Conjuntos

da planta e, então, desenvolver seu próprio sistema de classificação que lhes permitisse montar o banco de dados.

Há um outro pacote popular, concebido para crianças menores, o Your Facts ("Seus Fatos"). Pergunta-se à criança seu nome, se é menino ou menina, se tem algum animal de estimação, relógio de pulso, bicicleta, irmãos ou irmãs. Quando as informações sobre uma criança terminam de ser introduzidas, o programa pergunta se alguém mais gostaria de entrar com seus dados. Há espaço para fatos sobre quarenta crianças e, assim, pode-se incluir uma classe inteira. Quando todas as informações tiverem sido introduzidas e "Não" for digitado em resposta a "Alguém mais gostaria de entrar com seus dados?", as crianças retornam ao menu. Podem então ver todos os registros, verificar quais delas caíram em determinada categoria — como a das que têm relógio ou a das que têm bicicleta — ou podem brincar com um videojogo. Neste, o computador tenta acertar o nome da criança, depois de fazer perguntas como "Você é um menino?", "Você tem relógio?", e assim por diante. Então pergunta, por exemplo, "Seu nome é Tiago?". O Your Facts revela-se uma introdução ideal, para crianças pequenas, ao conceito de banco de dados. Refere-se a elas pessoalmente, encoraja a leitura e a escrita e é um exercício divertido.

Informação classificada
A construção e a pesquisa em bancos de dados relacionam-se a outros trabalhos feitos por crianças nas escolas.
O programa de matemática moderna introduz a idéia de conjuntos desde tenra idade. A teoria dos conjuntos diz respeito à classificação de dados e às relações entre cada classificação. Montar um banco de dados é uma atividade análoga a essa, e consiste em classificar as propriedades de um item em campos e registros, de forma que o banco possa ser pesquisado para fornecer as relações entre os itens.
Em particular, os conceitos de interseção e subconjunto têm importante significado na pesquisa do banco de dados, correspondendo à criação de sinopses. A interseção é a sobreposição de dois conjuntos — por exemplo, a interseção do conjunto de crianças louras com o de crianças que têm bicicletas é o conjunto de crianças louras que têm bicicletas.

O programa One World

Uma empresa australiana produtora de software, a Active Learning Systems, produziu um extenso banco de dados com informações sobre quase todos os países do mundo. O programa chama-se One World ("Um Único Mundo") e inclui indicações aos professores e planilhas para a classe e para os alunos individualmente. Contém mais de trinta itens de informação sobre cada país, incluindo fatos a respeito do sistema político, importações, exportações, línguas, limites, religiões, taxa de alfabetização, fatos históricos e tratados. Dá as porcentagens das populações rural e urbana, assim como a porcentagem da força de trabalho nos setores de produção, manufatura e serviços. Também fornece as proporções de áreas desérticas, cultivadas e florestais. O que torna o programa tão atraente, como ocorre com a maioria dos bancos de dados, são seus recursos de pesquisa e análise dos dados.

O menu principal dá uma série de opções:

1. Apresentar um país
2. Pesquisar com informações exclusivas
3. Analisar usando vários critérios
4. Auxílio à consulta
5. Saída

Se uma das crianças conhece um certo Mitterrand, um líder político europeu, e quer encontrar seu país, escolhe a segunda opção. O menu Dados de pesquisa permitiria à criança introduzir o nome Mitterrand em 3. Líder do governo e 4. Chefe de Estado. O programa então informaria que François Mitterrand é o chefe de Estado da França.

A opção Análise é o recurso mais poderoso do programa. Permite às crianças analisarem as informações, usando até três opções, a partir de vinte critérios. Por exemplo, poderiam determinar em quais países europeus a maioria da população vive no campo. Isso exigiria somente duas opções: região e porcentagem da população. Se as crianças ouvissem que os mineiros de bauxita no Suriname tinham entrado em greve, rapidamente descobririam a localização desse país, saberiam que a bauxita é seu principal produto de exportação e que a greve poderia ter efeito devastador sobre a economia.

O One World desenvolve a capacidade de pesquisa e interpretação de dados, encorajando as crianças a tirarem conclusões a partir da análise destes. É um grande apoio para as aulas de história, geografia e ciências sociais.

Um banco de dados fornece às crianças boa parte dos dados disponíveis e as estimula a pesquisar as respostas. Elas podem observar como o computador manipula as informações, entender a importância de fazer as perguntas corretas e organizar adequadamente suas entradas. Aquelas que, após experimentar o uso de um banco de dados, mais tarde entrarem para cursos superiores e para o mercado de trabalho certamente já possuirão importantes e relevantes habilidades.

Iniciação à geometria
Neste exercício os alunos iniciam-se nos princípios da geometria com a tartaruga Valiant. Eles participam ativamente na construção do labirinto e na programação da tartaruga, para fazê-la caminhar pelos obstáculos. O trabalho ensina os rudimentos do deslocamento angular e outros conceitos abstratos que as crianças costumam achar incompreensíveis e sem graça.

NOVA APRENDIZAGEM

Uma tartaruga mecânica auxilia o jovem estudante, na sala de aula, a entender conceitos abstratos de matemática. Além de divertidos, os robôs podem representar um eficiente método de ensino.

O uso de robôs na educação é uma tentativa de solucionar o problema — premente nas crianças pequenas — da compreensão de conceitos abstratos sem o correspondente modelo concreto. Complexas noções de trigonometria, por exemplo, podem ser demonstradas fisicamente; representam-se os processos mentais pelos movimentos reais de uma tartaruga tridimensional.

A geometria sempre foi ensinada com instrumentos tradicionais — lápis, régua e transferidor. Para uma criança de oito anos, aprender como medir um ângulo tem pouca importância, muito menos compreender os conceitos mais amplos da geometria. Por outro lado, os ângulos ganham um significado muito maior quando determinam se uma tartaruga, programada pelo aluno, irá ou não se chocar com uma floresta de garrafas plásticas dispostas no chão da sala de aula. Ainda que a geometria permaneça nebulosa, haverá uma motivação para entender suas aplicações práticas. A diferença entre distâncias de 15 e 20 cm fica rapidamente evidenciada quando resulta de uma escolha entre um ângulo de 5 ou 8 graus.

Um dos mais importantes aspectos da tecnologia das tartarugas está na possibilidade de aprendizagem através de tentativa e erro. A criança modifica a programação até conseguir um traçado satisfatório, enquanto vai adquirindo a compreensão das propriedades dos ângulos, retas e medidas. A ocorrência de um erro significa apenas que algo mais precisa ser tentado, até que ocorra o movimento desejado, não que algo tenha fracassado.

A programação de um braço mecânico para levantar e mover um objeto requer um raciocínio nos três planos do movimento; imaginar esse deslocamento no papel, que é bidimensional, não se mostra muito fácil. Com um braço robótico, no entanto, os planos do movimento são visualizados com clareza. Os resultados da programação aparecem, facilmente se detectam seus erros, e as alterações apropriadas podem ser introduzidas até que a mancha transcorra perfeita. O robô fornece uma realimentação instantânea ao programador, demonstrando cada ângulo e cada medida em seqüência, de maneira muito mais eficiente que um exercício tradicional com lápis e papel. Isso também permite ao aluno avaliar seu próprio trabalho.

Um brinquedo robótico popular em escolas infantis (de jardins de infância à primeira série) é um tanque futurístico chamado Big Trak, dotado de uma pequena memória e um teclado na parte superior. Digitando-se as setas e mais um número de 1 a 99, comandam-se os movimentos do tanque para todas as direções. As teclas numéricas permitem ainda indicar a distância a ser percorrida. Podem ser armazenados até oito comandos de cada vez. Utiliza-se esse brinquedo como substituto da tartaruga, encorajando as crianças a programarem rotinas simples. O Big Trak demonstra que os comandos são executados na ordem em que foram dados — uma útil iniciação à programação de computadores.

Objeto que ajuda a pensar

Embora a tartaruga possa auxiliar no ensino da matemática, essa não é sua principal função; ela foi projetada para ensinar a criança a programar um computador. O idealizador da tartaruga, Seymour Papert, acha que as crianças devem programar o computador, em lugar de serem por ele programadas. Essa filosofia é evidente na linguagem LOGO, desenvolvida por Papert para fins educacionais. Ele descreve a tartaruga como "um objeto para pensar", com o qual uma criança aprende os princípios básicos da programação utilizando a linguagem LOGO para controlar as ações do robô.

Enquanto a maioria dos projetistas se concentrava no desenvolvimento de hardware, Paul Cheung, da Universidade de Edinburg, sentiu a necessidade de um software mais versátil, e desenvolveu os aspectos robóticos do LOGO. Cheung desejava que as crianças pudessem controlar uma variedade de robôs. O LOGO é ideal para programar tartarugas, mas se mostra limitado para programar outros dispositivos.

Tomando o LOGO como ponto de partida, Cheung reestruturou a linguagem, permitindo o controle de mais funções e aumentando as possibilidades de receber sinais de entrada a partir de vários dispositivos, como sensores de toque, leitoras de código de barras e fotossensores. O resultado foi o Concurrent-LOGO, ou C-LOGO, uma linguagem capaz de realizar processamento "paralelo", isto é, executar diversas funções simultaneamente. O C-LOGO pode controlar até oito processadores de cada vez.

Duas funções particularmente interessantes são FOREVER ("para sempre") e WHENEVER ("sempre que"). A primeira, FOREVER, inicia um determinado processo que só se encerra pelo C-LOGO. A WHENEVER aguarda que surja uma determinada condição e, então, aciona uma função específica; é essencial para o processamento paralelo. O C-LOGO permite ligar e desligar chaves e verificar se alguma está ligada ou desligada. Utiliza-se esse recurso com o sensor — se um sensor de toque encosta num obstáculo, liga uma chave, informando ao computador que o contato foi realizado. Uma criança, utilizando a função WHENEVER, pode programar um

carrinho para evitar obstáculos, ordenando o desvio do robô-carrinho, cada vez que houver contato com algum obstáculo. O C-LOGO é mais adequado para os alunos do segundo grau, onde Paul Cheung o utilizou com diversos dispositivos de controle.

Alunos de treze a quinze anos desenvolveram programas para acionar os botões que controlam os movimentos das portas e janelas de uma casa de brinquedo automatizada. Essa casa tinha alarme contra roubos, ativado pela abertura de uma janela, e portas que se abriam somente após a digitação de uma senha no computador. Os dois subprogramas eram executados simultaneamente por processamento paralelo.

Os três pavimentos da casa eram servidos por um elevador, programado a fim de parar em cada andar. Para um funcionamento apropriado, o computador precisava saber em que andar estava o elevador. Este, construído com um brinquedo mecânico de armar, era acionado por um motor de corrente contínua e três chaves de contato, que serviam para detectar se o elevador estava em determinado andar. As crianças o programaram para descer ao térreo quando se pressionasse um botão; um outro botão enviava-o para o segundo andar; e um último, ao terceiro. As crianças se perguntaram: "O que acontecerá se um botão for acionado quando o elevador estiver em movimento?" — sua primeira resposta foi programar o elevador para ignorar todos os sinais até a parada. Mas, pensando melhor, decidiram que uma solução mais adequada seria o computador armazenar as instruções e implementá-las em seqüência.

Soluções simples

As características do C-LOGO permitiram desenvolver, de forma simples, um programa que sem ele seria bem mais complexo. Foram definidos como componentes necessários um detector de sinais, um seqüenciador e um controlador de elevador: o detector de sinais e o controlador do elevador agiam como processos paralelos. Os sinais eram detectados com FOREVER e transferidos para o seqüenciador. O controlador do elevador enviava uma mensagem ao seqüenciador, solicitando um destino: movia-o para o andar correto e solicitava novo destino. Após armazenar as instruções na ordem correta, o seqüenciador as enviava ao controlador. Empolgadas com o fato de o elevador estar na sua frente, assim como os meios de controlá-lo, as crianças persistiram até a resolução do problema.

Muitos educadores acreditam que um dos benefícios decorrentes da programação de robôs por crianças está no conceito de erro, que muda radicalmente. A experiência em tentativa e erro — e o conseqüente desenvolvimento de um método mais seqüencial de planejamento — traz o certo ou errado para um âmbito mais realístico, com correções constantes e mudanças para adaptação a novas situações.

Brincando com a tartaruga

O papel da tartaruga na sala de aula não se limita à geometria, trigonometria e programação. Já se construíram labirintos como parte de um projeto sobre mitologia (Teseu e o Minotauro) e sobre arquitetura de igrejas (o labirinto no chão da catedral de Reims). Um ambicioso projeto envolvia a construção, nas aulas de trabalhos manuais, de uma cidade em miniatura, que a tartaruga deveria percorrer.

Outros jogos com tartarugas envolviam pistas de corrida, em que cada criança programava a tartaruga, competindo com outras crianças. No jogo do Carteiro, sentadas em círculo, elas enviam mensagens umas para as outras, levadas pela tartaruga.

Dois jogos desenvolvidos especialmente para ajudar a ensinar o deslocamento angular e a distância são o Gira-tartaruga e o Empurra-tartaruga. No primeiro, posiciona-se a tartaruga no centro de um círculo dividido em seções; cada seção é marcada com um número diferente (equivalente aos pontos) e a criança precisa programar a tartaruga para girar e apontar para uma das seções; vence quem conseguir a contagem mais alta com o menor número de comandos no programa. O Empurra-tartaruga utiliza um método semelhante, mas com traçados retos.

Trajetória programada
O Big Trak é um brinquedo que pode ser programado de modo similar ao LOGO, utilizando o seu teclado numérico. Os comandos são constituídos de uma direção (direita, esquerda etc.) seguida de um número (até 99). O Big Trak pode armazenar e executar seqüencialmente até oito diferentes comandos.

PACOTES FINANCEIROS

Um empresário poupará tempo e dinheiro ao automatizar sua indústria, se usar, em lugar de uma equipe especializada para desenvolver programas, um pacote de software já pronto.

A primeira idéia que ocorre a um pequeno ou médio empresário que decide automatizar sua empresa é comprar um microcomputador e contratar um programador para desenvolver seus programas. Poucos aprenderam que esse não é o melhor caminho.

Antes de mais nada, o empresário deve se conscientizar dos problemas existentes em sua indústria e conhecer suas reais necessidades de automação. Quando não for possível decidir sozinho, deverá recorrer a uma empresa especializada, que fará uma análise e traçará as diretrizes a seguir.

Diferente do que o usuário em geral pensa, o microcomputador não é a parte mais importante na automação comercial. Antes de pensar no equipamento, o empresário deve se preocupar com os softwares que mais correspondam às suas necessidades. E, para obtê-los, pode seguir dois caminhos: desenvolver seus próprios programas (através de um programador ou uma equipe especializada) ou comprar pacotes (programas prontos, disponíveis no mercado).

Pacotes de programas

Todo empresário, ao iniciar a automação de sua indústria, deve começar pelo departamento cujos problemas requerem soluções mais urgentes. Geralmente a necessidade de decisão é mais imediata no setor contábil.

Como a rotina da área contábil e financeira é muito parecida na maioria das empresas, algumas software houses desenvolvem pacotes de programas para facilitar o contato direto entre computadores e usuário. São de fácil manipulação e não necessitam de operadores especializados. O próprio empresário e seus funcionários podem utilizá-los sem problemas.

Existem várias sotware houses que, além de fornecerem os pacotes, também assessoram as empresas que querem automatizar seus departamentos. Uma delas é a Tecnosoft Informática, que atua há muitos anos no mercado prestando serviços a grandes indústrias. Na área financeiro-contábil, a Tecnosoft desenvolve três tipos de pacotes destinados a equipamentos de portes diferentes: o bureau de serviços destinado aos minicomputadores IBM; o pacote para micros, compatível com as linhas Cobra 210 e 305; e o pacote destinado aos mínis Cobra 480 e 500, que são equipamentos mais sofisticados.

O último dos três é operado com terminal on line, pelo próprio usuário. Não sofre interferência do centro de processamento de dados e dispensa o custo operacional. Trata-se de um sistema orçamentário com previsão antecipada de doze meses; emite relatórios mensais confrontando os valores orçados com os valores reais.

Outra característica desse sistema é o gerenciamento de todo o departamento, emitindo, a qualquer momento, a posição de cada conta, com os lançamentos atualizados, apresentando assim um fechamento real ou fictício.

O pacote compatível com a linha Cobra 210 é um pouco mais simples de processar e também pode ser operado pelo usuário através de menus de opções, que facilitam o trabalho para principiantes. Este, assim como o pacote para o Cobra 500, emite diário-razão, balancetes sintético e analítico, bem como catálogos de contas e estoques. Por suas características simplificadas, dirige-se mais para pequenas e médias empresas.

Diferentemente do pacote para a linha 210, o bureau de serviços visa mais a grande empresa. Sua abrangência equivale à dos demais sistemas.

Nos três sistemas, os totais são impressos de cima para baixo, permitindo melhor visualização das tabelas e apuração dos percentuais de participação em relação a cada conta de primeiro nível, isto é, a relação entre ativo, passivo, receita e despesa.

Outra empresa especializada em pacotes de sistemas é a MCS - Micromática Computadores e Sistemas, que também comercializa equipamentos. O mais recente lançamento da empresa, o Placon (Planejamento, Programação e Controle de Obras) destina-se à área de construção ci-

vil, mas também se adapta perfeitamente a outros ramos de atividades.

O sistema do Placon voltado ao setor financeiro identifica os custos de uma obra, faz uma projeção dos índices que vão atuar sobre cada um dos insumos e avalia, mensalmente, quais os gastos referentes à construção. Além disso, faz um cronograma econômico para prever o momento certo de utilização dos recursos. Com ele é possível calcular, ainda, fluxo de caixa, contas a pagar e receber, saldos bancários e toda a parte contábil.

Outro sistema de bastante aceitação nas empresas de médio e grande porte é o Portfólio, utilizado no planejamento e controle de investimentos em ações e mercado a termo. Permite ao analista financeiro decidir com segurança quanto ao melhor momento de comprar ações, vendê-las ou fazer aplicações em commodities e em mercadorias.

Um módulo do sistema Portfólio faz análise econômico-financeira de empresas. Dessa forma, uma companhia interessada em aplicações na área financeira pode ter um perfil das empresas de seu interesse, antes de decidir pela compra de papéis de uma delas. Logicamente não será apenas esse perfil que pesará na hora da decisão. O feeling comercial do empresário também é parte importante no processo, mas os dados fornecidos pelo software darão enorme respaldo à escolha. Todos os sistemas são desenvolvidos pela MCS para equipamentos de 8 e 16 bits.

Vantagens dos pacotes

Desenvolver programas não é privilégio de firmas especializadas. Qualquer empresário poderá ter seus próprios softwares, produzidos por um programador ou por uma equipe contratada para esse fim. Mas, ainda assim, estará correndo sérios riscos. Além de estar desembolsando uma soma — às vezes acima de seu orçamento —, existe a possibilidade de ficar insatisfeito com os resultados finais.

Podemos atribuir esses detalhes à relativa facilidade em se desenvolver um programa. Muitos programadores principiantes, dispostos a ganhar algum dinheiro extra, propõem-se a desenvolver programas para empresários com pouco ou nenhum conhecimento de informática. Já outros empresários, julgando ter bastante conhecimento da área, arriscam-se a desenvolver seus próprios programas, sem ajuda de profissionais. Os resultados, nesse caso, podem ser ainda mais desastrosos e os prejuízos decorrentes, incalculáveis.

Para um empresário com pouco ou nenhum conhecimento de informática, ou mesmo alguém com algum esclarecimento, mas cauteloso na hora de investir seu dinheiro, o indicado são os pacotes de sistemas. Embora tenham forma quase generalizada, os pacotes mostram-se flexíveis e adaptáveis à rotina de trabalho da maioria das empresas. As próprias software houses se encarregam de adaptar os sistemas às particularidades das empresas. Em geral, essas adaptações são feitas nos gráficos e relatórios, que raramente coincidem em duas ou mais empresas.

Todas essas particularidades têm outra função que o usuário nem sempre percebe: proteger as informações da empresa. Por serem quase padronizados, os pacotes apresentam características que os diferenciam entre si, para dificultar o acesso de pessoas estranhas aos dados da empresa. A Tecnosoft, por exemplo, fornece ao usuário uma senha secreta de acesso, a fim de proteger seus programas e informações.

De fácil entendimento, o pacote dispensa o uso de pessoal especializado e tem seu preço reduzido. Além disso, apresenta a vantagem de ser incorporado sem dificuldade à rotina de trabalho da empresa. Pode-se implantar o Placon, da MCS, por exemplo, numa empresa não automatizada e que necessite de um sistema complexo, em trinta dias. E os usuários levarão de seis a oito meses para dominar o processamento, o que é um período relativamente curto.

Como implantar o sistema

Existem vários estágios de conhecimento da área de informática. Na maioria das vezes, o empresário menos esclarecido se vê ludibriado por vendedores interessados apenas nas comissões, ou por pessoas de má fé, com o intento de prejudicar o funcionamento de sua empresa. Por isso, o industrial interessado em automatizar sua empresa deve procurar informações e esclarecer dúvidas junto a empresas idôneas especializadas em assessoria nessa área.

Uma das informações prioritárias que o usuário deve procurar obter antes de se lançar definitivamente no campo da informática é quanto à utilidade do computador e como ele pode ser adaptado às necessidades da empresa. Muitos executivos, ao pensar na automação de suas empresas, procuram cursos de computação para aprender as diversas linguagens fugindo de seus propósitos iniciais — o custo e os benefícios do equipamento. Um erro comum é pensar que o aprendizado do BASIC elementar bastará para desenvolver programas complexos. Estes exigem sofisticados recursos de programação, muitas vezes incorporando rotinas em ASSEMBLY.

Uma empresa de consultoria, além de orientar o empresário, fará um diagnóstico da empresa e estabelecerá suas prioridades em relação ao sistema. Isso porque estas podem não recair na área financeira. Dependendo do ramo de atividade da empresa, outros departamentos poderão ter mais urgência de automação.

Essa análise pode ser um simples diagnóstico ou então um plano diretor, diferenciando-se apenas pela abordagem dos detalhes. Em ambos os casos, os resultados são satisfatórios. Para uma grande empresa, com sistemas de automação mais complexos, vale a pena um plano diretor; já na pequena e média empresa, um diagnóstico é suficiente, pois, além de menor custo, usa linguagem acessível aos principiantes.

EXPERT

O Expert apresenta design moderno, amplos recursos e um conjunto de periféricos próprios. Produto da empresa brasileira Gradiente, segue o padrão MSX, de grande sucesso no mercado internacional.

Num esforço conjunto para padronizar o hardware e o software dos micros de 8 bits, mais de vinte fabricantes japoneses e europeus adotaram o padrão MSX (Microsoft Extended), que determina a configuração básica do micro, embora permita variações nos detalhes. No Brasil, os primeiros micros nesse padrão surgiram em 1985. A Gradiente, que ocupa privilegiada posição como fabricante brasileira de equipamentos de som, lançou nesse ano o Expert, com índice de nacionalização em torno de 80%.

Sua CPU é um processador Z80A de 8 bits, que opera em conjunto com dois processadores auxiliares, dedicados a vídeo e áudio. Essa arquitetura — considerada a mais avançada na área de 8 bits —, aliada a uma elevada freqüência de clock (3,58 MHz), proporciona ao Expert alta velocidade de processamento. Sua RAM, de 80 Kbytes, destina 16 ao processamento de vídeo e 64 ao usuário. A ROM, de 32 Kbytes, contém o sistema operacional e o interpretador MSX-BASIC, versão mundialmente aceita dessa linguagem e virtualmente idêntica à utilizada no IBM PC e compatíveis. Ambas as memórias, de leitura e de acesso aleatório, permitem expansão para até 256 Kbytes, por meio de cartuchos.

O teclado do Expert separa-se da unidade principal, para maior conforto na utilização. Há 89 teclas, incluindo o conjunto completo dos caracteres da língua portuguesa e um bloco numérico separado, tipo calculadora. Acima e à esquerda, encontram-se cinco teclas de função, que, em conjunto com a tecla [Shift], produzem até dez diferentes funções. À direita destas, há cinco teclas especiais, de grande utilidade na programação: [Stop], [Home], [Select], [Insert] e [Delete]. Ainda seguindo o padrão MSX, as quatro teclas de movimentação do cursor são de tamanho grande e dispõem-se num conjunto separado, para fácil operação.

A Gradiente desenvolveu para o Expert toda uma família de periféricos, harmonizados com seu moderno design e que otimizam sua utilização. O monitor de vídeo monocromático, de formato quadrado, tem controles externos de contraste, brilho, fundo invertido e redução de imagem. Apresenta 40 colunas de 24 linhas e resolução de 256 x 192 pixels. A Gradiente oferece também um adaptador para tevê comum; a saída RGB, para monitor colorido, encontra-se incorporada ao Expert.

Para o armazenamento de dados e programas, a Gradiente desenvolveu o Datacorder, um minicassete especial que lê e grava dados com rapidez e segurança. O Expert, porém, pode utili-

EXPERT

MICROPROCESSADOR

Z80A.

CLOCK

3,58 MHz.

MEMÓRIA

80 Kbytes de RAM e 32 Kbytes de ROM.

VÍDEO

Monitor monocromático.
Modo texto: 24 linhas de 40 caracteres.
Modo gráfico: resolução de 256 x 192 pixels.

TECLADO

Tipo QWERTY, com 89 teclas, com caracteres da língua portuguesa e um bloco numérico separado.

LINGUAGENS

MSX BASIC.

PERIFÉRICOS

Saída RGB, impressora padrão Centronics; Datacorder; gravadores cassete; joysticks e interface opcional para comunicações.

DOCUMENTAÇÃO

Dois manuais — um a respeito do equipamento e outro do MSX BASIC — mais uma cartela contendo o resumo de operação do Expert.

zar qualquer gravador cassete; e, inversamente, o Datacorder é compatível com qualquer micro. Para programas aplicativos, existem duas entradas de cartucho na unidade central. Os usuários de disquetes deverão aguardar o lançamento de uma unidade de discos Gradiente compatível com o padrão MSX.

Na traseira da unidade central ficam as saídas para acoplamento de periféricos. Além da saída RGB, há uma para impressora e duas para joystick; o Expert aceita o de fabricação Gradiente ou qualquer outro padrão Atari. Além dessas conexões embutidas, a Gradiente oferece uma interface opcional para comunicações que acopla até oito diferentes expansões, inclusive um modem. Com isso, o Expert torna-se um terminal de videotexto com acesso a bancos de dados públicos, como o Cirandão, da Embratel.

Já existe grande quantidade de programas desenvolvidos para o padrão MSX no mercado internacional. A Gradiente tem, também, convênios com diversas software houses brasileiras para lançamento de programas aplicativos. Além dos utilitários básicos, a empresa investe muito no setor educacional. No Brasil, a exemplo do que ocorre no Japão e na Europa, a "cultura do MSX" vem abrindo novas possibilidades aos usuários de micros de 8 bits.

Painel frontal

Saída de áudio

Conexão data corder (cassete)

Controle de volume

Saída de vídeo composto monocromático

Entrada de joystick

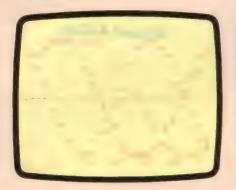
Entrada de cartuchos

Interruptor
Alto-falante
RAM do usuário (64 Kbytes)
Saída para impressora
Cabo de alimentação
Tomadas de força chaveadas
Processador
(áudio, AY -3 - 8910)
Porta-fusível
Conector para
teste da placa
Expansão bus MSX
Conexão do teclado
RAM de vídeo (16 Kbytes)
Controle de endereçamento
e teclado
Cristal
Saída de vídeo
Comutação (120 - 240 V)
Painel traseiro
Processador
(vídeo, TMS9128)
Placa analógica
EPROM
Microprocessador

BIORRITMO

Este programa, para linha Sinclair, vai lhe permitir traçar seu biorritmo considerando os ciclos físico, emocional e intelectual.
É um bom recurso para melhor dirigir seus esforços.

"Tudo flui e reflui, com movimento pendular. Tudo cresce e decresce; tudo ascende e descende; tudo caminha e volta ao ponto de partida". Essa frase, do dr. Krumm-Heller, é a síntese do biorritmo; ou seja, o ritmo que rege a fisiologia e a mente dos seres vivos. O programa a seguir determina as fases do biorritmo humano conforme seus três ciclos: físico, emocional e intelectual. Há ainda a opção gráfica, que traça na tela o biorritmo para os próximos trinta dias e o copia na impressora. Para reiniciar, aperte qualquer tecla.

```
10 REM
BIORRITMO
PARA LINHA SINCLAIR
20 CLS
30 REM   PEDE DADOS
40 GOSUB 710
50 INPUT D1
60 PRINT D1,,"MES=";
70 INPUT M1
80 PRINT M1,,"ANO=";
90 INPUT A1
100 PRINT A1,,,,"ENTRE A DATA D
E NASCIMENTO:","DIA=";
110 INPUT D0
120 PRINT D0,,"MES=";
130 INPUT M0
140 PRINT M0,,"ANO=";
150 INPUT A0
160 PRINT A0,,,,
170 REM   CALCULA N.° DE DIAS
180 LET A=A0
190 LET D=D0
200 LET M=M0
210 GOSUB 740
220 LET DI=DF
230 LET A=A1
240 LET D=D1
250 LET M=M1
260 GOSUB 740
270 LET DEL=DF-DI
280 REM   IMPRIME NA TELA
290 PRINT "DIAS VIVIDOS=,DEL,,
,"CICLOS="
300 PRINT
310 LET A$="FISICO:"
320 LET B$="EMOCIONAL:
330 LET C$="INTELECTUAL:"
340 LET D$=" - - > + "
350 LET E$=" - - >    "
360 LET P=23
370 GOSUB 800
380 LET W=X
390 LET P=28
400 GOSUB 800
410 LET Y=X
420 LET P=33
430 GOSUB 800
440 LET Z=X
450 LET DEL=DEL+1
460 LET P=23
470 GOSUB 800
480 IF X>W THEN PRINT A$;W;D$
490 IF X<W THEN PRINT A$;W;E$
500 PRINT
510 LET P=28
520 GOSUB 800
530 IF X>Y THEN PRINT B$;Y;D$
540 IF X<Y THEN PRINT B$;Y;E$
550 PRINT
560 LET P=33
570 GOSUB 800
580 IF X>Z THEN PRINT C$;Z;D$
590 IF X<Z THEN PRINT C$;Z;E$
600 PRINT
610 PRINT AT 21,6;"GRAFICO ? (S
/N)"
620 INPUT K$
630 IF K$="S" THEN GOTO 840
640 CLS
650 STOP
660 GOSUB 710
670 PRINT D1;"/";M1;"/";
680 GOTO 100
690 CLS
700 STOP
710 CLS
720 PRINT TAB 11;"BIORRITMO";AT
2,0;"ENTRE A DATA DESEJADA:","D
IA=";
730 RETURN
740 IF M>=3 THEN GOTO 770
750 LET A=A+1
760 LET M=M+12
770 LET M=M+1
780 LET DF=INT (A*365.25)+INT (
M*30.6)+D
790 RETURN
800 LET X=DEL/P
810 LET DF=X-INT X
820 LET X=INT (SIN (2*DF*PI)*10
0)/100
830 RETURN
840 REM   GRAFICO
850 CLS
860 PRINT "QUAL O SEU NOME ?"
870 INPUT N$
880 PRINT N$
890 PRINT "DE A PRIMEIRA DATA:"
,"DIA=";
900 DIM S(12)
910 LET S(1)=31
920 LET S(2)=28
930 LET S(3)=31
940 LET S(4)=30
950 LET S(5)=31
960 LET S(6)=30
970 LET S(7)=31
980 LET S(8)=31
990 LET S(9)=30
1000 LET S(10)=31
1010 LET S(11)=30
1020 LET S(12)=31
1030 INPUT D1
1040 PRINT D1,,"MES=";
1050 INPUT M1
1060 PRINT M1,,"ANO=";
1070 INPUT A1
1080 PRINT A1
1090 LET DEL=365.25*A1+30.6*M1+D
1-365.25*A0-30.6*MØ-DØ
1100 LET S(2)=S(2)+(A1/4=INT (A1
/4))
1110 PAUSE 120
1120 CLS
1130 SLOW
1140 PRINT TAB 5;"-  B I O R R I
T M O -";TAB 7;"
1150 LET D1=D1-2
1160 LET E=D1+1
1170 PRINT AT 5,0;"+";TAB 0;"1";
AT 19,0;"-";TAB 0;"1";AT 12,0;"0
..................................."
1180 FOR G=D1 TO D1+30
1190 LET DEL=DEL+1
1200 LET E=E+1
1210 IF E=S(M1)+1 THEN GOSUB 139
0
1220 LET P=23
1230 GOSUB 800
1240 PRINT AT 12-8*X,1+G-D1;"F"
1250 LET P=28
1260 GOSUB 800
1270 PRINT AT 12-8*X,1+G-D1;"E"
1280 LET P=33
1290 GOSUB 800
1300 PRINT AT 12-8*X,1+G-D1;"I"
1310 IF (G-D1)/5=INT ((G-D1)/5)
THEN PRINT AT 21,G-D1;E
1320 NEXT G
1330 REM   COPIA NA IMPRESSORA
1340 COPY
1350 LPRINT N$
1360 PAUSE 4E4
1370 FAST
1380 RUN
1390 LET E=1
1400 LET M=M+1
1410 IF M=13 THEN LET M=1
1420 RETURN
```

O caminho
O diagrama mostra três caminhos possíveis através do labirinto, percorridos graças a três diferentes estratégias de busca: aleatória, heurística e sistemática. Neste exemplo, o método heurístico atinge o objetivo mais rapidamente que os outros, embora seja possível projetar um labirinto em que isso não aconteça. Na prática, os problemas envolvendo labirintos costumam ser solucionados por uma combinação dos métodos heurístico e exaustivo (aquele em que todos os pontos são considerados).

Heurístico

Sistemático

Aleatório

BUSCA INTELIGENTE

Encontrar o caminho num labirinto é um problema clássico da inteligência artificial, exigindo avançadas técnicas de busca. Mostramos aqui três dessas técnicas, incorporadas num único programa computacional.

Imagine três ratos num labirinto, onde existe um pedaço de queijo. Um deles cambaleia durante alguns minutos e então adormece (o insensível experimentador colocou gim em sua água). O segundo rato é mais metódico: apóia contra a parede a pata esquerda dianteira e prossegue, virando à esquerda a cada bifurcação e retrocedendo quando encontra um beco sem saída. Termina por atingir o objetivo — mas, a essa altura, o terceiro rato já comeu o alimento. Este, de olfato mais aguçado, periodicamente farejava o ar e avançava pelo caminho que pressentia estar conduzindo-o para mais perto do delicioso cheiro. Sua estratégia de busca revelou-se, sem dúvida, a mais inteligente das três.

A busca é um conceito-chave no campo da inteligência artificial. Seu filho pode estar no quintal brincando de caça ao tesouro enquanto você, comandante sentado em sua poltrona, faz palavras cruzadas: tanto você quanto ele estão procurando alguma coisa. O exemplo dos ratos no labirinto mostra-se proveitoso, visto que muitos problemas diferentes podem ser abordados por meio de técnicas de busca. De fato, os três ratos ilustram diferentes tipos:

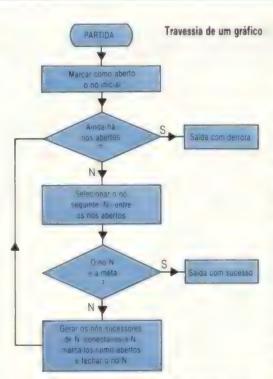
- Busca aleatória (o "passeio do bêbado");
- Busca exaustiva (enumeração sistemática);
- Busca heurística (exploração dirigida).

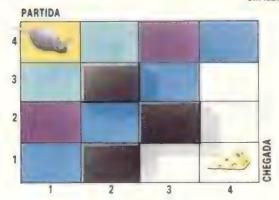
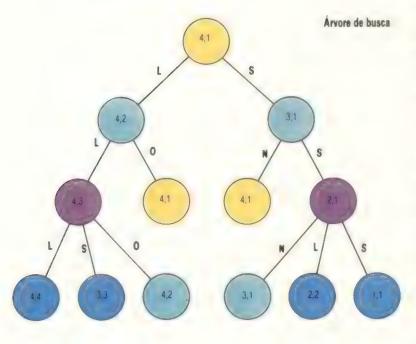
A terceira estratégia é considerada uma abordagem mais inteligente do problema porque, em geral, exige menos esforço para atingir a solução. Todos os métodos heurísticos, porém, devem contar com algum meio de saber quando a busca se aproxima do objetivo (por exemplo, o sentido do olfato no rato). Sem esse conhecimento, não se pode agir de modo inteligente: será preciso confiar na enumeração sistemática das possibilidades, ou esgotar todas elas.

Achar o caminho num labirinto é um típico exercício de busca. (Muitos problemas não espaciais, entretanto, tais como a integração de uma expressão matemática simbólica, podem ser abordados com o mesmo referencial de busca.)

Busca ramificada

A maioria dos métodos de busca empregados pela inteligência artificial envolve a geração de uma árvore, criada a partir do esquema do labirinto a ser atravessado. Na árvore aqui mostrada, toma-se o ponto de partida como raiz e ramificam-se os níveis seguintes mediante a consideração de todos os movimentos possíveis que partem do nó inicial. Movimentos em diagonal não são permitidos. O fluxograma aponta o algoritmo básico para a geração da estrutura da árvore.

Travessia de um gráfico

Um labirinto simples

Árvore de busca

Existem dois meios para se julgar a qualidade de um método de busca:

- Qual o tempo necessário para se encontrar um caminho?
- Esse caminho é o mais econômico?

Idealmente, queremos encontrar a solução ótima no menor prazo possível. Na prática, porém, podemos ser forçados a aceitar uma solução subótima, sob pena de gastarmos demasiado tempo até encontrar a melhor solução.

As estratégias de busca utilizadas pela inteligência artificial são, em sua maioria, elaboradas de acordo com um plano comum, que possibilita atribuir pesos distintos aos critérios de desempenho. O diagrama de algoritmo genérico relativo a travessia de gráfico *(ao lado)* mostra uma família de métodos de busca. Escolhe-se um deles definindo-se o preenchimento do retângulo central: qual o próximo nó a ser examinado? Os termos "aberto" e "fechado" referem-se ao estado de qualquer nó do percurso. Os abertos são candidatos a exame; os fechados já foram examinados. Os primeiros ficam numa lista de espera — e o segredo da eficiência está na ordem de processamento dessa lista.

Pesquisa em árvore

A busca se realiza com a construção de uma árvore. O labirinto de quatro retângulos por quatro *(à esquerda)* apresenta o rato-robô no canto superior esquerdo e o queijo no inferior direito; de cada retângulo, o rato pode ir para norte, sul, leste ou oeste, embora alguns (os escuros) estejam bloqueados. Se traçarmos as opções existentes no início do percurso, veremos uma estrutura de busca em árvore.

A partir do retângulo inicial (4,1), o rato pode mover-se para leste (4,2) ou para sul (3,1). Saindo de (4,2), pode ir para leste ou para oeste; e, de (3,1), vai para norte ou para sul. Caso se mova para leste e depois para oeste (ou para o sul e depois para o norte), voltará ao ponto de partida, o que não será uma decisão muito inteligente. Isso mostra, porém, que o mesmo retângulo pode aparecer em diferentes nós da árvore, correspondentes aos vários caminhos para se chegar até eles.

A "busca exaustiva", um método sistemático de explorar a árvore, investiga os nós em ordem de proximidade ao ponto de partida, ou raiz da árvore. Assim, considera cada seqüência de N movimentos (no nível N da árvore) antes de qualquer seqüência de N + 1 movimentos. Objetiva encontrar um caminho que exija o menor número de passos; como todos os passos são igualmente dispendiosos, o caminho será uma solução de dispêndio mínimo — que, entretanto, pode demorar a ser encontrada. À medida que se multiplicam as ramificações da árvore, o tempo necessário para se atingir o objetivo aumenta exponencialmente.

Aperfeiçoar uma busca exaustiva requer uma fonte de informações heurísticas relativas à distância do objetivo. Tomemos como exemplo um

bairro onde os quarteirões sejam quadrados exatos. Para irmos de A até B devemos andar um certo número de quarteirões para norte ou sul, e outros tantos para leste ou oeste. Da mesma forma, no labirinto, um rato inteligente pode calcular a quantos retângulos está do alvo; sabendo quando a busca se aproxima do final, ele pode andar mais depressa.

O rato do início deste artigo seguia uma regra simples: chegar cada vez mais perto da meta. É a estratégia "subir o morro", assim chamada por ser comparável à escalada de uma montanha envolta na névoa: basta mover-se sempre para cima. Pode ser bem mais rápida do que a busca exaustiva, mas não garante que se encontre o caminho ótimo.

O método chamado A-asterisco (A*) consiste num compromisso meio-termo entre "subir o morro" e a busca exaustiva. O nó a ser examinado é selecionado com base na soma HD + DP, onde HD representa a estimativa heurística da distância e DP a distância percorrida. Quanto mais correta a estimativa de HD, mais eficaz a busca. Se houver erro em HD, convém que seja por subestimar a distância e não por superestimá-la.

Neste programa-exemplo, utilizamos a estrutura de um só programa para implementar os três métodos, com pequenas alterações. As únicas diferenças são evidenciadas na escolha do nó examinado em seguida:

Busca exaustiva — escolha a menor DP;
Subir o morro — escolha a menor HD;
Algoritmo A* — escolha a menor HD + DP.

As linhas do labirinto
O labirinto está contido numa matriz bidirecional M (,), e as estruturas de dados para a árvore de busca, contidas nas matrizes P( ), S( ), N( ) e H( ). O programa combina os fatores custo-até-aqui e custo-daqui-até-o-objetivo, que podem ser ponderados alternando-se os valores W1 e W2 nas linhas 1080 e 1090. Com os valores da listagem, a estratégia vigente é do tipo "subir o morro"; o método heurístico utilizado baseia-se no exemplo dos quarteirões simétricos. Fazendo-se prevalecer o método heurístico (com W2 > W1), acelera-se a busca.

Programa de procura no labirinto

```
1000 REM ******************************
1010 REM ** RELATORIO 2.1            **
1020 REM ** PROGRAMA DE PROCURA NO   **
1030 REM ** LABIRINTO                **
1040 REM ******************************
1050 MH = 17:MW = 25: REM ALTURA E PROFUNDIDADE DO LABIRINTO
1060 SI = 256: REM LIMITES DA ARVORE
1070 WA = 1:RR = 2:FO = 3:DN = 4:BL = 5
1080 W1 = 1: REM PESO PARA SF
1090 W2 = 2: REM PESO PARA HD
1100 DIM M(MH + 1,MW + 1): REM O LABIRINTO
1110 DIM C$(5): REM CARACTERISTICAS DO LABIRINTO
[illegible]
1130 REM CAMINHO,PASSOS,NO,DISTANCIA HEURISTICA
1140 REM
1150 REM --RATO NO LABIRINTO:
1160 GOSUB 1360: REM FAZER O LABIRINTO
1170 NC = 0: REM NUMERO DE NOS EXAMINADOS
1180 K = 0: REM CONTADOR
1190 GOSUB 1660: REM LIMPE TODOS OS CAMINHOS
1200 N(1) = 2 * MW + 2: REM PRIMEIRO NO ABERTO
1210 S(1) = 0:H(1) = FR - 1 + (FC - 1)
1220 P(1) = 0: REM SEM PREDECESSOR
1230 REM --LOOP PRINCIPAL:
1240 REM ****LOOP PRINCIPAL****
1250 GOSUB 1770: REM ESCOLHE PROXIMO NO S
1260 NC = NC + 1
1270 VT 1 22: PRINT NC,SR,SC,H(S);"
1280 GOSUB 1880: REM GERA SUCESSOR
1290 IF FR < > SR AND NC < 300 THEN 1240
1295 IF FC < > SC AND NC < 300 THEN 1240
1300 VTAB 23: PRINT "PROCURA TERMINADA"
1310 IF FR = SR AND FC = SC THEN GOSUB 2290: REM RETRACE OS PASSOS
1320 IF NC > 300 THEN PRINT "FALHOU"
[illegible] PRINT NC;"NOS EXAMINADOS."
1340 END
[illegible]
1360 REM ROTINA PARA CRIAR O LABIRINTO:
[illegible] FOR P = 1 TO MH + 1
1380 FOR R = 1 TO MW + 1
1390 IF RND (1) < 0.28 THEN M(P,R) = WA
1395 IF RND (1) < 0.28 THEN M(P,R) = BL
1400 IF P = 3 OR R = 3 THEN M(P,R) = BL
[illegible]
[illegible] IF P > MH OR R > MW THEN M(P,R) = WA
1430 NEXT R: NEXT P
1440 FR = 2 + INT ( RND (1) * (MH - 1)): REM FILEIRA DE COMIDA
1450 FC = 4 + INT ( RND (1) * (MW - 3)): REM COLUNA DE COMIDA
1460 M(FR,FC) = FO
1470 M(2,2) = RR: REM RATO ROBO
1480 C$(BL) = " "
1490 C$(WA) = CHR$ (88): REM MANCHA
1500 C$(DN) = " "
1510 C$(RR) = "R"
1520 C$(FO) = "F"
1530 GOSUB 1560: REM MOSTRE O
1540 RETURN
[illegible]
1560 REM --ROTINA PARA MOSTRAR O LABIRINTO
[illegible] HOME
1580 FOR R = 1 TO MH + 1
1590 FOR C = 1 TO MW + 1
1600 HTAB C: VTAB R: PRINT C$(M(R,C));
1610 NEXT C
1620 NEXT R
1640 RETURN
[illegible]
1660 REM --ROTINA PARA LIMPAR A ARVORE:
1670 DD = 9999: REM MORTO
1680 FOR Q = 1 TO SI
1690 P(Q) = 0
1700 S(Q) = DD
1710 N(Q) = 0
1720 H(Q) = DD
1730 NEXT Q
1740 NN = 2: REM PROXIMO NO LIVRE
1750 RETURN
1760 :
1770 REM --ESCOLHE O MELHOR NO S
1780 S = 1:BN = DD
1790 FOR I = 1 TO SI
1800 V = S(I) * W1 + ABS (H(I)) * W2
1810 IF V < BN AND H(I) > = 0 THEN S = I:BN = V
1820 NEXT I
1830 IF S = 1 THEN 1835
1835 HTAB 0: VTAB 20: PRINT "EXPLORANDO...."
1840 SR = INT (N(S) / MW)
1850 SC = N(S) - MW * SR
1860 RETURN
1870 :
1880 REM --ROTINA PARA GERAR SUCESSORES:
1890 IF H(S) = 0 THEN RETURN : REM FEITO
1900 REM --NORTE
1910 Y = SR - 1:X = SC
1920 IF Y > 1 THEN GOSUB 2090
1930 REM --ESTE
[illegible]
1950 IF X < = MW THEN GOSUB 2090
1960 REM --SUL
1970 Y = SR + 1:X = SC
1980 IF Y < = MH THEN GOSUB 2090
1990 REM --OESTE
2000 Y = SR:X = SC - 1
2010 IF X > 1 THEN GOSUB 2090
2020 REM --FECHA TAMBEM O NO S
2030 H(S) = - H(S)
2040 IF H(S) > 0 THEN PRINT "NOSSA"
2050 HTAB SC: VTAB SR: PRINT " "
2060 M(SR,SC) = DN
2070 REM CELULAS BRANCAS NA TELA
2080 :
2090 REM ROTINA PARA ABRIR UM NO
2100 IF M(Y,X) = DN THEN RETURN
2105 AW = WA
2110 IF M(Y,X) = AW THEN RETURN
2120 REM --PRIMEIRO ACHE A LOCALIZACAO LIVRE
2130 NI = 0
2140 REM **LOOP PARA LOCALIZAR
2150 IF S(NN) < > DD THEN NI = NI + 1:NN = NN + 1
2160 IF NN > SI THEN NN = 1
2170 IF NI > SI THEN PRINT "ESTA CHEIO!": STOP
2180 IF S(NN) < > DD THEN 2140
2190 REM --AGORA ABRA O
2200 IY = X + Y * MW
2210 N(NN) = IY
2220 P(NN) = S
2230 S(NN) = S(S) + 1
2240 H(NN) = ABS (Y - FR) + ABS (X - FC)
2250 HTAB X: VTAB Y: PRINT "+"
[illegible] REM O MOSTRA NA TELA
[illegible] RETURN
[illegible]
2290 REM ROTINA PARA CAMINHO DE VOLTA
2300 ST = S(S)
[illegible] FOR Q = 1 TO 10000: NEXT Q: GOSUB 1560
[illegible] HTAB FC: VTAB FR: PRINT C$(FO);
2330 REM **IMPRIMA O CAMINHO**
2340 S = P(S): REM NO PARENTE
2350 IY = N(S): REM COORDENADAS
[illegible]
[illegible]
2380 M(Y,X) = RR: REM PEGADA DE RATO!
2385 IF X < > 0 OR Y < > 0 THEN 2390
[illegible]
[illegible] HTAB X: VTAB Y: PRINT "*"
[illegible] IF S > 0 THEN 2330
[illegible] HTAB 2: VTAB 2: PRINT C$(RR)
[illegible] VTAB 20: PRINT "CAMINHO DE";ST;"PASSOS"
[illegible] PRINT NC;"NOS FECHADOS."
2440 RETURN
```

O programa foi escrito para os micros da linha Apple. A versão para o TK 2000 é igual à da Apple. Na versão para o CP-500 e compatíveis TRS, trocar as seguintes linhas:

Apple e TK 2000:
Na listagem ao lado, eliminar a linha 1230 e substituir na linha 1520 'F' por 'Q' (queijo em vez de food)

CP-500 e TRS:
Trocar as seguintes linhas:

```
1050 MH=9: MW=15
1270 PRINT @768, NC
1300 PRINT @832
     "PROCURA
1390 IF RND(0)<0.28 THEN
     M(P,R)=WA ELSE
     M(P,R)=BL
1395        [eliminar]
1440 ...RND(O)...
1450 ...RND(O)...
1600 PRINT @ R*64+C,
     C$(M(R,C)),
1830 IF S=1 THEN PRINT
     @ 704,
     "EXPLORANDO"
1835        [eliminar]
2050 PRINT @ SR*64+SC,
     " "
2250 PRINT @ Y*64+X,
     "+";
2320 PRINT @ FR*64+FC,
     C$(FO),
2390 PRINT @ Y*64+X, "*"
2410 PRINT @ 130, C$(RR),
2420 PRINT @ 704,
     "CAMINHO DE"; ST,
     PASSOS
```

ROTINAS GENÉRICAS

Para tornar rotinas em ASSEMBLY relocáveis, de modo a serem executadas qualquer que seja sua localização na memória, utilizamos símbolos e rótulos no lugar de endereços e valores absolutos.

O ASSEMBLY é uma linguagem de programação de baixo nível, ou seja, compõe-se de comandos "primitivos", que a CPU processa um a um. Por isso, o programador vê-se obrigado a escrever e reescrever rotinas para realizar tarefas essenciais, que já se encontram incorporadas aos comandos de uma linguagem de alto nível — gerenciamento de entrada e saída, por exemplo, ou operações aritméticas com 2 bytes. A melhor solução é montar uma biblioteca — em disco, fita ou papel — com as rotinas mais usadas e anexá-las aos novos programas quando necessário.

Esse processo, contudo, envolve dois grandes problemas. O primeiro é a dificuldade de se escreverem rotinas importantes, e às vezes longas, genéricas o suficiente para funcionarem em diferentes programas sem adaptações. O segundo consiste em escrever rotinas que não residam permanentemente num único local da memória, mas possam ser reposicionadas, por meio de uma nova compilação, com um endereço ORG diferente, continuando ali a desempenhar as mesmas funções.

Ambos os problemas derivam da questão da generalidade e compatibilidade, também existente no BASIC, e as soluções também são as mesmas utilizadas pelos programadores dessa linguagem: usar variáveis para passar valores do programa principal para uma sub-rotina; usar nas sub-rotinas variáveis locais, de modo a torná-las independentes do programa principal; evitar o uso de valores absolutos, sejam numéricos ou alfabéticos, bem como números de linhas.

Já nos habituamos a considerar os endereços de memória em ASSEMBLY como equivalentes às variáveis em BASIC; isto é, os comandos atuam sobre o conteúdo dos endereços, da mesma forma que um programa em BASIC manipula o conteúdo das variáveis. Até agora, em geral, nos referimos às posições de memória pelo seu endereço absoluto — de início um hábito conveniente, mas que devemos abandonar em nome da generalidade. Em vez de endereços e valores absolutos, passaremos a usar símbolos; em vez de variáveis e número de linhas, usaremos uma série de pseudocódigos operacionais. Já vimos exemplos de ambos os tipos, como este programa:

6502			Z80		
DATA1	EQU	$12	DATA1	EQU	$12
DATA2	EQU	$79	DATA2	EQU	$79
	LDA	DATA1		LD	A,(DATA1)
LOOP	ADC	DATA2		ADC	A,(DATA2)
	BNE	LOOP		JR	NZ,LOOP
	RTS			RET	

Temos aqui dois tipos de símbolos — dois valores e um rótulo — usados como operandos de instruções em ASSEMBLY. Esse recurso torna a rotina genérica e também relocável. As únicas quantidades absolutas são os valores de DATA1 e DATA2, que podem inicializar-se no programa principal, e não no início da própria rotina.

Há outros pseudocódigos à nossa disposição — DB ("definir byte"), DW ("definir palavra") e DS ("definir armazenamento"). (Note que, assim como ORG e EQU, esses nomes podem variar de um compilador para outro.) Os três comandos permitem inicializar e alocar posições de memória, como neste exemplo:

			ORG	$D3A0
D3A0	5F	LABL1	DB	$5F
D3A1	CE98	LABL2	DW	$98CE
D3B3		LABL3	DS	$10
D3B3		DATA1	EQU	LABL3

TABELA DE SÍMBOLOS
LABL1 = D3A0: LABL2 = D3A1: LABL3 = D3A3
DATA1 = D3A3
ASSEMBLY COMPLETO - NÃO HÁ ERRO

Nessa listagem completa em ASSEMBLY, produto do programa compilador, vemos, na parte inferior, uma tabela que associa os símbolos utilizados no programa aos valores que representam. Há várias coisas importantes a observar. A linha que começa com LABL1 utiliza o pseudocódigo DB. Vemos na listagem que o comando ORG atribuiu a LABL1 o endereço $D3A0, o que a tabela de símbolos confirma. O efeito de DB aqui é colocar o valor $5F no byte endereçado por LABL1; assim, a posição de memória $D3A0 inicia com o valor $5F, como vemos na coluna do código de máquina.

Note também que LABL2 representa o endereço $D3A1; contudo, como DW inicializa uma "palavra", isto é, 2 bytes consecutivos, o valor $98CE ficará armazenado nas posições $D3A1 e $D3A2 sob a forma baixo-alto (lo-hi), o que também se vê na coluna "Código de máquina". Como DW sempre converte seus operandos na forma baixo-alto, usa-se com freqüência esse pseudocódigo para inicializar posições indicadoras, isto é, que apontam para outros endereços. A posição LABL2, ou $D3A1, enquadra-se nessa categoria, pois aponta para o endereço $98CE.

Observe ainda que a instrução DS $10 acrescenta $10 ao contador do programa. Isso fica mais claro na tabela de símbolos que na própria listagem: LABL3 representa a posição $D3A3 (que se segue à instrução anterior), embora na listagem esse valor pareça ser $D3B3. Este último é o endereço da instrução que se segue a DS; assim, DS $10 reservou um bloco de 16 bytes (de $D3A3 a $D3B2 inclusive) entre as duas instruções. Esse processo assemelha-se à inserção de longas linhas REM num programa em BASIC a fim de criar um espaço não utilizado na área reservada ao texto do programa, para então processá-la com PEEKs e POKEs, como se fosse uma área de programa em código de máquina.

A última instrução do programa utiliza EQU, que dá a um símbolo o mesmo valor de outro; assim, DATA1 assume o valor $D3A3, atribuído a LABL3. Aqui pode haver uma confusão: LABL3 é a representação simbólica do endereço $D3A3; portanto, a instrução DATA1 EQU LABL3 significa "o símbolo DATA1 passa a assumir o mesmo significado e o mesmo valor do símbolo LABL3". O fato de a instrução DB ter colocado o conteúdo $5F na posição $D3A3 em nada altera o significado dos símbolos LABL3 e DATA1. Aqui nos deparamos com uma das maiores dificuldades do aprendizado básico do ASSEMBLY: separar claramente os conceitos de "endereço" e "conteúdo do endereço". Talvez, ao aprender BASIC, você tenha esbarrado na mesma dificuldade de separar as variáveis de seus conteúdos, que são os valores que elas assumem.

À primeira vista, o comando DB parece uma duplicata de EQU, mas essa impressão é falsa. LABL1 aqui significa "a posição $D3A0", e a instrução DB $5F atribuiu o valor $5F a esse byte; contudo, embora o valor de LABL1 já esteja estabelecido, o conteúdo da posição que ele representa pode alterar-se a qualquer momento do programa (por exemplo, armazenando ali o conteúdo do acumulador). Da mesma forma, DATA1 é um símbolo cujo valor estabeleceu-se pela instrução EQU; esse valor não se altera pela execução do programa. Também assim, LABL3 indica o início de uma área de dados de 16 bytes, cujo conteúdo se altera durante a execução do programa; o próprio LABL3, no entanto, é inalterável.

			ORG	$D3A0
D3A0	4D4553	LABL1	DB	'MESSAGE 1'
D3A9	CE98	LABL2	DW	$98CE
D3BB		LABL3	DS	$10
D3BB		DATA1	EQU	LABL3

TABELA DE SÍMBOLOS
LABL1 = D3A0: LABL2 = D3A9: LABL3 = D3AB
DATA1 = D3AB
ASSEMBLY COMPLETO - NÃO HÁ ERRO

Essas são algumas das possibilidades de ação dos pseudocódigos; há outras, ainda. Considere a seguinte nova versão da rotina anterior: A instrução DB tem como operando o string 'MESSAGE1', e o compilador inicializou os endereços de $D3A0 a $D3A8 com os valores ASCII dos caracteres entre aspas. Isso transparece na coluna de endereços, e se confirma pela coluna "Código de máquina", onde se vê que o conteúdo dos 3 bytes de $D3A0 a $D3A2 é, respectivamente, $4D, $45 e $53 — a notação hexadecimal dos códigos ASCII relativos a "M", "E" e "S".

Esse é um recurso importante, pois dispensa o programador da tarefa de converter mensagens e dados alfabéticos em listas de códigos ASCII, e também facilita a leitura da listagem. Além disso, sugere a possibilidade de se obterem saídas em tela a partir de programas em ASSEMBLY. Isso é relevante, pois até agora nos restringimos a armazenar resultados na memória e verificá-los usando programas especiais. No decorrer do curso, abordaremos a manipulação da tela, mas antes devemos estudar outros aspectos do ASSEMBLY. Contudo, não é difícil compreender co-

EXERCÍCIOS

1) A primeira rotina citada no texto utiliza o pseudocódigo DS para reservar $10 bytes de memória a partir do endereço representado pelo rótulo LABL1. Escreva um programa em ASSEMBLY para armazenar nesse bloco os números de $0F a $00, em ordem decrescente, um número por byte. Para tanto, use um loop e técnicas de endereçamento indexado; você necessitará das instruções DEX (decrementar o registrador X) ou DEC (IX + 0) (decrementar IX). O loop deve continuar decrementando o registrador de índice, mas sem chegar a ativar o flag zero. Use as instruções de desvio BNE ou JR NZ.

2) A partir do segundo programa citado no texto e utilizando as técnicas empregadas no exercício anterior, escreva um programa que copie a mensagem armazenada em LABL1, pelo pseudocódigo DB, num bloco de memória iniciando no endereço armazenado por DW, no rótulo LABL2. Se o endereço $98CE não for adequado ao seu computador, mude a inicialização; mas o programa deverá funcionar para qualquer endereço, e para mensagens de qualquer tamanho. Para tanto, seu programa deve usar o número de caracteres da mensagem como contador de um loop, ou então deve identificar o final da mensagem. Você pode colocar, por exemplo, um asterisco como caractere final de qualquer mensagem.

mo um programa endereça a tela, partindo do princípio de que sempre se armazenam resultados na memória, e de que a RAM de tela é também uma simples área de memória endereçável.

O recurso mais importante oferecido pelo pseudocódigo DB é que ele torna LABL1 semelhante a uma variável alfanumérica do BASIC. Quando escrevemos em BASIC

```
200 LET A$ = 'MESSAGE 1'
```

na realidade estamos criando um ponteiro que indica o início de uma tabela de bytes contendo os códigos ASCII de "M", "E", "S" etc. Cada vez que o interpretador BASIC encontra uma referência a A$, ele procura em sua própria tabela de símbolos o endereço apontado, isto é, a posição inicial do conteúdo de A$. Da mesma forma, em nosso programa em ASSEMBLY consideramos, LABL1 como equivalente de A$, pois já escrevemos uma rotina que manipula uma tabela por meio do endereçamento indexado.

Vemos assim que os pseudocódigos nos permitem eliminar dos programas os endereços e valores absolutos, substituindo-os por símbolos; isso torna as rotinas mais compatíveis e relocáveis. O que nos falta agora é um meio de acessar esses módulos transportáveis a partir do programa principal. Em outras palavras, precisamos de um equivalente do comando GOSUB do BASIC.

Esse equivalente existe: é a instrução JSR, na versão do 6502, ou CALL na do Z80. Ambas requerem como operando um endereço absoluto (que pode ser um rótulo), e colocam esse endereço em lugar do conteúdo do contador do programa. Portanto, a próxima instrução a ser executada será a primeira da sub-rotina assim endereçada. A execução continua, a partir desse ponto, até encontrar a instrução equivalente a RETURN — seja RTS ou RET. Esta irá substituir o conteúdo atual do contador do programa pelo endereço que lá estava imediatamente antes da chamada da sub-rotina por meio da instrução JSR ou CALL. Assim sendo, a próxima instrução a ser executada será aquela que vem logo em seguida a JSR ou CALL. Esse mecanismo é idêntico ao utilizado pelo interpretador BASIC ao executar uma sub-rotina por meio de um GOSUB, e dela retornar com um RETURN. Facilmente se entende isso tudo, mas surge o problema de como restaurar o conteúdo anterior do contador do programa após a execução do RETURN. A resposta é que as instruções JSR e CALL primeiro colocam o conteúdo do contador numa pilha, antes de substituí-lo pelo endereço da sub-rotina; por sua vez, a instrução RTS ou RET extrai esse endereço da pilha e o devolve ao contador. No próximo artigo do curso veremos em mais detalhe o que é essa pilha, e como a manipulamos.

Instruções essenciais

EDISA

Acompanhando a evolução
A Edisa desenvolveu uma linha de equipamentos capazes de se adaptarem à evolução das empresas, como o multiusuário/multitarefa ED-281

A Edisa é uma empresa voltada estritamente para computadores tanto de médio como de grande porte. Seu ED-680 é o primeiro equipamento nacional a empregar o processador Motorola 68000.

Como quase todas as empresas nacionais de computadores, a Edisa - Eletrônica Digital S.A. também teve início na década de 70. Por intermédio de um contrato de cooperação técnica com a empresa Fujitsu, do Japão, a Edisa começou sua produção de minicomputadores no país em novembro de 1977. A partir daí, sua preocupação tem sido desenvolver e aperfeiçoar equipamentos para atender à crescente automação industrial, comercial e, mais recentemente, bancária. Atualmente, além de sua linha de minicomputadores, a Edisa produz também micros e supermicrocomputadores.

Um ano após a fundação da empresa, seu controle acionário passou às mãos do grupo financeiro Iochpe, tradicional no setor bancário, mas que também atua nas áreas agrícola e de papel. Em 1980, a Edisa começou a fabricar os equipamentos ED-300 — até então apenas montados no país. Nesse mesmo ano teve início a produção dos sistemas de entrada de dados ED-100. Em 1981, a empresa conseguiu mais uma fatia do mercado, passando a fazer os terminais financeiros ED-3400, destinados à automação bancária.

Na área de recursos humanos, a Edisa possui um setor de desenvolvimento destinado a dar oportunidade aos novos profissionais de informática e engenharia.

Esse apoio à educação é dado também fora da companhia, através de acordos com universidades, porém mais voltado para a área de software, onde os alunos desenvolvem programas subsidiados pela empresa. A produção de software decorre paralela à criação de novos equipamentos na empresa, possibilitando a atuação permanente de estudantes do ramo.

Os protocolos — ou periféricos destinados à comunicação entre dois ou mais tipos de processadores — também despertam interesse nas universidades. Assim, as escolas desenvolvem um projeto de periférico e o apresentam à empresa. Se houver interesse em sua comercialização, a Edisa firma contrato com a universidade e investe no desenvolvimento do aparelho.

Muitas escolas superiores possuem bons laboratórios e técnicos altamente qualificados, dispondo-se a desenvolver programas para a Edisa. Entre as universidades contratadas pela empresa estão a Federal do Rio Grande do Sul e a Federal do Rio de Janeiro.

A fábrica da Edisa localiza-se no Distrito Industrial de Gravataí, Rio Grande do Sul, numa área de 36.000 m². A empresa possui ainda filiais nas cidades de Porto Alegre, São Paulo, Rio de Janeiro, Brasília, além de escritórios nas principais cidades brasileiras.

A Edisa preocupa-se com a tecnologia avançada dos equipamentos estrangeiros. Exemplo disso é o ED-680, primeiro supermicrocomputador nacional a utilizar o processador Motorola 68000, que vem sendo usado há apenas dois anos no mercado internacional.

Outra característica da empresa consiste na atualização versátil de seus equipamentos, de forma a não tornar obsoletos os aparelhos adquiridos por seus clientes. Essas mudanças também estão presentes no ED-680, em sua nova versão, que utiliza discos Winchester com capacidade de armazenamento de 72 Mbytes por unidade, embutidos no próprio gabinete da CPU e acondicionados em espaço magnético selado.

A Edisa produz também o ED-381, um minicomputador que pode ser conectado a equipamentos de grande porte da linha IBM ou Burroughs, ou a equipamentos que utilizam as linguagens universais. Com memória real de 24 Kbytes a 256 Kbytes, esse míni desempenha funções em lotes ou on line.

A área comercial é a mais cobiçada pelas indústrias de informática. A cada dia surgem novos modelos de equipamentos versáteis, projetados de forma a se adaptarem à evolução da empresa, ou sistemas práticos para automatização de firmas comerciais em curto espaço de tempo. Voltada para esse campo, a Edisa criou a linha ED-200. O multiusuário/multitarefa ED-281 tem a capacidade de incrementação de acordo com o crescimento da empresa. Permite que várias pessoas executem diversas tarefas ao mesmo tempo, nos múltiplos setores da companhia. O ED-251 trabalha em 8 ou 16 bits. É um microcomputador desk-top, com processador Z80A e CPU com 64 Kbytes. Utiliza discos flexíveis de 5 1/4 polegadas.

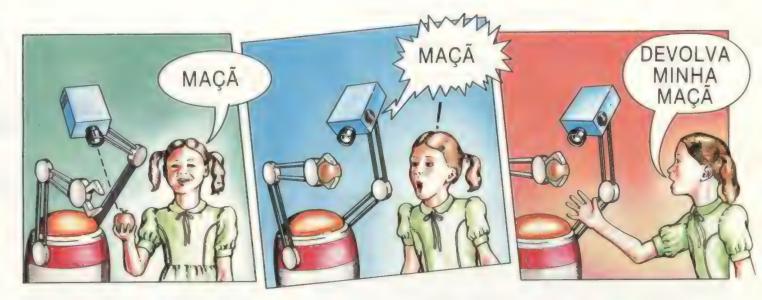
A Edisa também está voltada ao desenvolvimento de softwares para suprir seus equipamentos. Além dos programas elaborados pelas universidades, a empresa dispõe de pessoal especializado para atender a esse mercado, além de prestar serviços de consultoria aos clientes, a fim de orientá-los na aquisição dos sistemas mais adequados a suas atividades.

Mesmo os aplicativos desenvolvidos pela empresa possuem flexibilidade para serem adaptados às necessidades de cada cliente e ao perfil de sua empresa. A série ED-800 é uma família de sistemas interativos compatíveis entre si desenvolvida pela Edisa. O modelo ED-848 atende às necessidades de uma empresa de médio porte, mas tem possibilidade de expansão. Opera também como nodo principal em uma rede de sistemas distribuídos, através do software de comunicação de dados. Comporta até 152 usuários e pode ser processado em lotes ou on line.

O sistema ED-868 é um pouco mais sofisticado, dirigido a empresas de maior porte. Tem configuração de memória de 4 Mbytes, podendo expandir-se até 16 Mbytes. Apresenta capacidade para atender às necessidades de processamento centralizado ou distribuído para até quatrocentos usuários.

A Edisa oferece treinamento a seus clientes para possibilitar maior intimidade com os equipamentos. Esse treinamento também é oferecido aos funcionários das assistências técnicas autorizadas para manutenção dos equipamentos.

Solução de porte
O minicomputador ED-381 pode ser conectado a computadores de grande porte ou a qualquer outro que utilize linguagens universais.

CONVERSA DE ROBÔ

Dotar um robô de voz revela-se tarefa das mais difíceis, pois ainda não compreendemos a fundo o aprendizado da fala. Mas algumas teorias importantes podem esclarecer muitas questões nesse sentido.

O estudo da fala humana deu origem a duas linhas de pensamento: uma crê que a linguagem é inata; a outra, que é adquirida, ou aprendida. Os defensores da primeira hipótese afirmam ser o homem a única criatura a se comunicar por meio da linguagem; os da segunda citam experiências com animais ensinados para se comunicarem com os seres humanos por meio da linguagem de sinais.

Se de fato aprendemos a falar pela simples exposição à fala, justifica-se um método que leve os robôs ao mesmo resultado. Afinal, tudo ficaria mais fácil se ele aprendesse a linguagem apenas ouvindo-a em uso.

Houve algumas tentativas de expandir o conhecimento gramatical do computador através de exemplos adicionais de estruturas de sentenças; outras procuraram dar ao robô a capacidade de aprender novas palavras e morfemas (elementos da linguagem) em qualquer idioma pela simples exposição. Até hoje, porém, nenhum dos sistemas criados conseguiu ensinar um robô a falar.

Assim, para fins práticos, a capacidade de fala do robô depende da tese que afirma ser a linguagem inata, não aprendida, devendo-se elaborar as regras lingüísticas e incorporá-las permanentemente ao robô, tal como se ele tivesse nascido com elas. Em termos gerais, esse procedimento se divide em duas fases: as análises sintática e semântica.

A análise sintática ocupa-se da gramática daquilo que se diz e decodifica a estrutura superficial da mensagem ou codifica-a numa forma gramatical pronta para transmissão pelo robô. O método mais comum é o uso da "árvore de análise", que decompõe ou compõe uma sentença a partir das várias partes da fala. Embora não seja fácil, essa tarefa vem se desenvolvendo satisfatoriamente.

A análise semântica oferece mais problemas, por envolver a elaboração do sentido da mensagem (quando o robô ouve alguém falar) ou a criação da mensagem que deve ser transmitida (quando ele quer falar com alguém). Ocorre que o significado depende do contexto em que a linguagem é falada, e isso também se aplica a todo o contexto da mensagem. Esse contexto abrange tanto o conhecimento sobre o estado do mundo enquanto se fala quanto o conhecimento que cada um dos interlocutores tem do outro.

Essa abordagem foi adotada em experiências realizadas pelo cientista em computação Terry Winograd, criador de um programa que possibilitou a um robô compreender o conteúdo da fala e agir sob instruções. Mas Winograd simulou em computador um robô que funcionava num mundo minuciosamente definido — no caso, alguns elementos básicos manipuláveis. O programa de Winograd, chamado SHRDLU, alcançou uma boa análise semântica, mas o mundo que ele podia entender era extremamente sim-

Ver para crer
Ao ver um objeto e lhe dar um nome — maçã, por exemplo —, o homem compreende o significado de "maçã". O robô, por sua vez, é capaz de reconhecer o objeto (fazendo-o corresponder a uma imagem interna) e de repetir o padrão sonoro que combina com maçã; mas não sabe que o objeto é uma fruta comestível e que, de fato, "pertence" ao homem — coisa que o ser humano compreende perfeitamente.

ples. Um robô que trabalhasse no caos do mundo real encontraria dificuldade bem maior em compreender o conteúdo da fala.

O uso proveitoso da linguagem pelos robôs exige uma mensagem entre o usuário e a máquina, e vice-versa. Para o robô é relativamente fácil falar, pois tudo o que ele venha a expressar encontra-se confinado nos restritos limites de seu conhecimento. A dificuldade está em compreender o que o homem diz, pois isso implica problemas de análise.

Chegou-se a pensar que dados falados recebidos pelos robôs poderiam ser analisados por análise sintática da entrada, e que isso revelaria o significado da mensagem. Trabalhos recentes, porém, demonstram a importância do conhecimento do mundo e do contexto em que a mensagem se insere. Essa abordagem conduziu a experiências de análise sintática do sinal falado para obter-se uma primeira suposição quanto a seu significado. Assim, pelo conhecimento do mundo e das coisas que venham a ser expressas, o robô repassa a análise sintática inicial com o propósito de obter aos poucos uma análise correta do que se expressou. Mas isso ainda está muito além da capacidade dos robôs.

A síntese da fala

O método mais simples de síntese de fala utiliza um gravador cassete, por meio do qual o robô reproduz uma mensagem humana. Talvez isso frustre um pouco a idéia que se tem da fala de um robô, mas é o ponto de partida de todos os sistemas de síntese de fala.

Vejamos quais as limitações desse método e de que modo se pode desenvolvê-lo. A limitação mais evidente está nas características de um gravador: é mecânico, caro, volumoso e sujeito a freqüentes avarias. Considerado isso, o próximo passo será converter a mesma mensagem à forma digital, de modo que possa ser armazenada num chip da memória do robô. Para tanto, usa-se um convetsor analógico/digital, no qual se empregam números para representar a forma de onda da fala, que está em contínua variação. É exatamente esse método que se usa na gravação digital de música — por exemplo, nos sistemas de discos compactos.

Esse método também apresenta inconvenientes. Um deles é que o sinal digitalizado ocupa muito espaço na memória. A gravação com disco compacto examina amostras do sinal acústico cerca de 44.000 vezes por segundo, com uma resolução em torno de 16 bits (ou seja, a amplitude da forma de onda é armazenada como um número de 16 bits, possibilitando a distinção de 2^{16} níveis, em que $2^{16} = 65.536$). Com esse sistema, cada segundo de gravação ocuparia 88.000 bytes de memória. Fica claro que uma mensagem falada excede a capacidade de armazenamento de qualquer computador. Essa taxa de amostragem, entretanto, é aplicável apenas à reprodução de sons de alta fidelidade. Um sistema simples poderia funcionar com uma resolução de 8 bits e uma taxa de 3.000 amostras por segundo, exigindo apenas 3 Kbytes de memória.

Entretanto, para liberar o máximo de espaço de memória, outras economias são necessárias. A lingüística descobriu que a linguagem falada pode ser convenientemente decomposta em unidades de fala chamadas fonemas. De modo geral, concorda-se que no total existem quarenta diferentes fonemas na maioria das línguas faladas, sendo possível, assim, armazenar informações acústicas exatas, indispensáveis para descre-

Apricot F1

Big Ears

Currah

Ouvir e falar
É relativamente fácil criar a fala do robô e do computador. O mercado oferece sintetizadores, como o Currah, até mesmo para uso nos micros de menor porte. Mas existem certas dificuldades de reconhecimento devido às variações no modo como os seres humanos pronunciam os sons das vogais e à capacidade de processamento e de memória necessária para manipular um vocabulário extenso. Sistemas como o Big Ears e o Apricot F1 dispõem de uma pequena série de comandos reconhecíveis incorporados, porém insuficientes para abranger mais que um número mínimo de operações.

ver cada um desses quarenta fonemas e utilizá-los como base para a fala do robô. As informações de fonemas em geral são mantidas com um chip sintetizador de fala fabricado comercialmente, cabendo ao robô apenas encadear tais fonemas para gerar a mensagem necessária. Esta geralmente é mantida numa cadeia de números de fonemas na memória do computador.

Para se programar a maioria dos sintetizadores de fala em uso, escreve-se a mensagem que o robô deve transmitir numa versão fonética. Assim, a mensagem "O que é isso?" poderia ser escrita "U kié içu?", o suficiente para o chip sintetizador produzir a cadeia correta de sons. Essa não é, exatamente, a mesma representação utilizada pela lingüística na descrição dos fonemas, mas seria suficiente para o robô.

Percebe-se, neste ponto, que o robô deixou de utilizar uma mensagem pré-gravada, passando a gerar sua própria mensagem. Graças a isso, dirá tudo o que lhe foi pedido sem necessidade de armazenar previamente toda a mensagem.

Assim, podemos tentar programar algumas regras gramaticais para que o robô diga algo inteiramente original. Com uma ressalva: o número de coisas diferentes que o robô quererá dizer será bastante limitado, não havendo necessidade de complicação — a menos que seja grande a curiosidade de ver os resultados possíveis.

Quem já ouviu alguma vez um sintetizador de voz no terminal de serviços de bancos mais modernos sabe que a qualidade da fala, embora compreensível, não é perfeita. Isso ocorre por dois fatores: primeiro, a inflexão de um fonema usado pelo homem varia de acordo com os fonemas anteriores e posteriores; segundo, o som geral da fala humana modifica-se conforme o significado pretendido. Por exemplo: "Quer fazer o favor de sentar-se?" e "Quer fazer o favor de sentar-se?" são duas mensagens escritas de forma idêntica, mas, faladas, terão efeitos bem diferentes. Imagine a primeira emitida por uma gentil anfitriã a seu convidado, e a segunda por um irritado professor a um aluno. Tentou-se captar tais entonações nos sistemas de síntese de fala, mas existem diferenças na aplicação, visto que o robô não conhece o significado das palavras que usa.

Reconhecimento

Ao se criar um sistema de reconhecimento de fala, um problema fundamental e que exige solução refere-se às coisas que se pode dizer a um robô e às inúmeras maneiras de exprimi-las. Pode-se abordá-lo inicialmente com o uso de gravação em fita daquilo que se pretende que o computador reconheça. Então, enquanto se fala, ele simplesmente examina todas as fitas gravadas em busca da que apresenta maior semelhança com a mensagem ouvida — é assim que muitos dos robôs reconhecem a fala. Armazenam alguns "gabaritos" internos de mensagens faladas e, quando se "conversa" com eles, simplesmente buscam o gabarito que melhor corresponda à mensagem recebida. Obtêm-se tais gabaritos treinando o robô, repetindo-lhe palavras ou frases, até que ele extraia uma "média" do que "ouviu". Esse método dá bons resultados quando a quantidade de coisas a dizer ao robô é pequena e elas são sempre ditas da mesma maneira ou quase — isto é, com robôs que respondem a comandos simples como "Para a frente", "Vire à esquerda", e assim por diante.

Esse, porém, é um problema relativamente simples, identificado como reconhecimento discreto da fala — cada item emitido é "discreto", ou seja, separado das outras mensagens por uma pequena pausa, durante a qual ocorre o silêncio. O verdadeiro problema surge quando se deseja que o robô faça uso de fala contínua, a fala normalmente usada pelos seres humanos. É interessante experimentar algo assim como "O poço secou". O resultado será mais ou menos "Upossussekô", com a mistura dos sons e palavras.

Ao ouvir falas alheias, as pessoas resolvem esse problema tentando supor o que o falante quer dizer (em geral uma tarefa não muito difícil) e utilizando a suposição para decodificar a mensagem. Para agir assim, porém, o robô teria de conhecer muito sobre a intenção e o significado possíveis — esta sim uma tarefa complexa.

De modo geral, a síntese da voz pelos robôs vem se tornando comum, embora ainda haja muito a fazer no sentido de melhorar a qualidade da fala. Árdua é a tarefa do aperfeiçoamento do reconhecimento da voz; e, quanto a isso, o que se conseguiu foi munir o robô de uma compreensão da fala equivalente à de um cão bem treinado, capaz de responder a comandos verbais, desde que pouco numerosos. Constata-se, porém, o enorme interesse na solução dos problemas relacionados à fala do robô, indicando, para um futuro próximo, consideráveis avanços.

Ordem sonora
O Voicemate, um braço mecânico controlado por voz, foi desenvolvido pelo Departamento de Ciências e Engenharia da Newcastle Polytechnich, para uso industrial e em laboratórios.

Antecipação
No início das pesquisas sobre inteligência artificial, a capacidade de um computador jogar xadrez era considerada a medida mais expressiva de sua potencialidade. Hoje, a maioria dos programas de xadrez possui recursos para antecipar uma série de movimentos do jogo, avaliando assim qual o melhor dos próximos lances. Para fazer isso, os computadores constroem uma árvore de busca.

GRANDE MESTRE

Os conceitos básicos por trás dos programas de xadrez geralmente envolvem "árvores" de busca. Vejamos alguns mecanismos que podem antecipar e determinar os melhores movimentos em jogos de estratégia.

As primeiras aplicações da inteligência artificial destinavam-se à resolução de problemas abstratos de matemática e de física. Paralelamente, porém, os pesquisadores — inclusive alguns pioneiros da computação, entre eles Claude Shannon, John von Neumann e Alan Turing — dedicaram-se à programação de máquinas para o xadrez, o jogo intelectual por excelência. Um programa bem-sucedido era considerado o teste supremo de inteligência de um computador.

Hoje existem sistemas computacionais com nível de jogo equivalente ao dos mestres internacionais — embora poucos afirmem que essas máquinas "pensem". Mesmo assim, o xadrez e outros jogos que exigem habilidade mental fornecem o terreno ideal para testar teorias de planejamento estratégico.

A maioria dos programas de jogos de habilidade apóia-se em técnicas de busca "em árvore", modificadas para levar em conta um adversário. A idéia fundamental é a de "antecipação". O programa desenvolve uma árvore de jogo considerando seus próprios movimentos, prevendo os possíveis contramovimentos do oponente, examinando antecipadamente suas respostas a eles e assim por diante.

A ilustração da página seguinte mostra a árvore de antecipação de um jogo imaginário entre MINI e MAX. A raiz da árvore é a posição atual, com MAX pronto para se mover. Os nós terminais, ou "folhas", são posições de final de jogo. Emprega-se a árvore para selecionar um movimento por meio de um processo conhecido como "minimaxização", claramente definido pela primeira vez em 1949, por Claude Shannon. De início, atribuem-se valores numéricos aos nós terminais — digamos, 1 para vitória, 0 para empate e -1 para derrota. Esses valores combinam-se à medida que avançamos na árvore, na suposição de que o jogador (MAX) sempre escolhe o maior valor, enquanto o oponente (MINI) sempre escolhe o menor, produzindo assim valores para os nós seguintes.

Neste exemplo, o valor da raiz é 0: o jogo deverá empatar, desde que ninguém cometa um erro. O movimento correto a esse nível será então M1, M3 ou M4, mas não M2. As regras que regem a ramificação e a geração de valores dos nós são determinadas pelas regras do jogo específico. Nos mais simples, como o jogo-da-velha, é possível apresentar a árvore do início até o final. Já o xadrez tem um "fator de ramificação" de cerca de 32; isso significa que existem aproximadamente 32 jogadas legítimas a partir de qualquer posição. Uma jogada antecipada em quatro lances (dois movimentos de cada lado) levaria a mais de um milhão de nós terminais. Essa explosão combinatória implica que programas de xadrez não podem ter seu desenrolar antecipado até o final da partida.

Em vez disso, esses programas costumam antecipar até onde podem e avaliar as posições aí encontradas. Tal procedimento exige método para decidir se os nós são favoráveis ou desfavoráveis; é a chamada avaliação estática, fonte inevitável de imprecisão, visto ser apenas uma estimativa do resultado final. A lógica do exame com certa antecipação, usando uma função de avaliação imperfeita, está na sua aplicação próximo ao final do jogo — e, de qualquer modo, será melhor que a ausência de verificação.

Num jogo de damas, por exemplo, poderíamos criar uma função de avaliação muito simples, com apenas quatro termos, baseada em:

D, vantagem de dama;
P, vantagem de peças;
M, diferença de mobilidade; e
C, controle do centro do tabuleiro.

Tais atributos podem ser calculados pelo exame do tabuleiro. Por exemplo, D = Dd-Do, sendo Dd as damas da defesa e Do, as damas oponentes.

As outras características refletem vários aspectos estratégicos do jogo. É melhor ter mais peças que o oponente (o perdedor termina sem peças); será útil ter mais movimentos disponíveis; as casas centrais possuem maior valor que as laterais. O programa deve, de alguma forma, combinar essas quantidades numa contagem geral.

Vamos estabelecer, por exemplo, que uma dama (D) vale três peças comuns (P), uma peça vale dois e meio movimentos adicionais (M) e um movimento simples vale duas vezes o controle de uma casa central (C). Nossa função de cálculo será:

$$V = 15D + 5P + 2M + C$$

(São usados pesos inteiros para acelerar o cálculo.)

Essa, entretanto, é uma função de cálculo muito rudimentar: o programa clássico de jogo de damas de Arthur Samuel, criado no início dos anos 60, empregava até 25 parâmetros. Os coeficientes neste exemplo também são bastante arbitrários. Um dos aspectos atraentes do programa de Samuel era que ele ajustava seus próprios pesos automaticamente, o que consistia numa forma rudimentar de aprendizado.

A idéia de atribuir valores numéricos a características de jogos e combiná-los numa soma ponderada para calcular o valor de qualquer posição vem mostrando sua importância desde a década de 50. A função de avaliação tem um papel semelhante ao da estimativa heurística da distância na solução de problemas por meio de busca.

Um programa que se limite a antecipar sempre e com a mesma profundidade acaba em situações difíceis. Isso porque algumas posições do jogo são "estáveis", enquanto outras são bastante "instáveis". No xadrez, é provável que o estado do jogo após a captura ou promoção de um peão seja instável ao extremo, pois uma recaptura pode ocorrer no próximo lance. Caso isso aconteça, num lance além do "horizonte" do programa, a avaliação estará seriamente comprometida.

Para atenuar esse efeito, a maioria dos programas não possui antecipação com profundidade fixa, mas sim uma medida da "estabilidade" que indica se determinada posição pode ser avaliada de modo confiável. Se for evidenciada instabilidade na posição, a pesquisa avança ainda mais. Nos jogos de xadrez e damas, isso acarreta o exame comparativo de longas seqüências de captura.

O algoritmo alfa-beta surgiu em 1967, no programa MacHack, de Greenblatt. Trata-se de uma sofisticação da minimaxização básica — fornece o mesmo resultado, mas com esforço muito menor. O diagrama da página seguinte mostra parte da árvore de jogo entre MINI e MAX.

Estado do jogo

O diagrama mostra a situação num jogo-da-velha após seis lances (o xis joga em seguida). Pode-se construir uma árvore simples para acompanhar os próximos três movimentos finais, pelo exame das opções abertas a cada estágio.
A primeira jogada possui três nós, referentes às casas 1, 2 ou 7, que podem ser ocupadas pelo xis. Quando o círculo joga, há somente duas casas para escolha, com três possibilidades (duas das quais estarão disponíveis após o xis ter jogado). O próximo lance, portanto, tem seis nós.
O final indica situações em que todas as casas foram ocupadas — caso o jogo não tenha se decidido — e consiste em nós "terminais".

Seleção das casas
Uma vez construída a árvore, cada nó terminal recebe um valor: 1 para a vitória dos xis, 0 para um empate e -1 para a vitória dos círculos. Podemos então percorrer a árvore em sentido contrário, atribuindo valores aos nós anteriores. Se tomarmos o nó mais à direita na primeira jogada, chegaremos ao valor -1 considerando os dois nós abaixo dele, de valores 0 e -1. Se for a vez dos círculos, o valor mínimo será escolhido, isto é, -1. Voltando pela árvore até a jogada atual, os xis deveriam escolher o maior valor disponível. Neste exemplo, o melhor que podem fazer é empatar a partida, pela escolha da casa 2 no próximo lance; as outras opções dariam a vitória aos círculos.

Os números dentro dos nós são avaliações. As letras em cada nó (de A até L) mostram a ordem em que se examina a árvore, a partir de um procedimento de primeira profundidade. A barra simples marca os cortes alfa e as barras duplas os cortes beta. Estes eliminam as ramificações que não podem afetar o resultado final.

O corte alfa na posição E significa que esse nó não precisa ser avaliado, bem como seus eventuais descendentes. Quando alcançamos o nó E, sabemos que o C obtém contagem 15; mas, na posição D, o oponente pode forçar-nos a descer para 10. Inútil verificar se há possibilidade de sermos forçados ainda mais para baixo, pois essa rota é evidentemente menos desejável que aquela que passa pelo nó C. Assim, os outros

Movimentos

1	d2-d3	e7-e5
2	Nb1-d2	Nb8-c6
3	g2-g3	Ng8-f6
4	Bf1-g2	Bf8-c5
5	e2-e3	d7-d5
6	Ng1-e2	0-0
7	a2-a3	Bc8-f5
8	b2-b3	Nf6-g4?!
9	h2-h3	ng4-f6
10	Bc1-b2	Qd8-d6
11	g3-g4	Bf5-e6
12	Ne2-g3	a7-a5
13	Qd1-e2	Nf6-d7
14	Ng3-f5	Be6xf5
15	g4xf5	Nd7-f6
16	h3-h4	Rf8-e8
17	Bg2-h3	a5-a4
18	b3-b4	Bc5xb4!?
19	a3xb4	Qd6xb4
20	Bb2-a3	Qb4xh4
21	Nd2-f3	Qh4-h5
22	Nf3-d2	Qh5xe2 + ?
23	Ke1xe2	b7-b5
24	c2-c3	h7-h6
25	Bh3-g2	Ra8-a5
26	Ra1-b1	Re8-b8
27	Rb1-b2	Rb8-b6
28	Rh1-b1	Kg8-h7
29	Ba3-c5	Rb6-b8
30	Bc5-a3	Nc6-a7
31	Nd2-f3	Nf6-d7
32	Nf3-e1	c7-c6
33	Nd1-c2	Rb8-a8
34	Nc2-b4	Ra8-d8
35	Nb4-a2	Na7-c8
36	c3-c4!	d5xc4
37	d3xc4	b5xc4
38	Bg2xc6	Nc8-a7
39	Bc6-e4	Nd7-f6
40	Ba3-e7	Rd8-c8
41	Be7xf6	g7xf6
42	Rb1-b6!	c4-c3
43	Rb6xf6	Kh7-g7
44	Rf6-b6	a4-a3
45	Rb1-g1 +	

Nesse ponto o programa estava indicando uma desvantagem de 2,4 peões, e os programadores optaram por abandonar a partida.

Posição das peças depois do lance 21

Este jogo de xadrez é parte de uma série em que o Cray Blitz — o mais poderoso programa de xadrez do mundo — foi derrotado (4 a 0) por David Levy, da Intelligent Software. Levy e os programadores do Blitz haviam apostado 5.000 dólares. Apesar de o programa ter alcançado o nível de mestre nacional, o jogo mostra que ainda há muito trabalho pela frente antes que um computador seja um adversário à altura dos melhores jogadores de xadrez.

descendentes do nó F podem ser eliminados da avaliação.

O mesmo raciocínio se aplica, inversamente, ao nó I. Quando chegamos a ele, verificamos que G tem uma contagem 20. O nó H, com 25, parece melhor — mas é MINI que escolhe entre os nós G e J, e sem dúvida irá preferir G. MAX nunca poderá chegar ao valioso nó I.

Podemos colocar essas idéias em termos de uma guerra de sexos. MAX, um machista, pensa ser o nó C, por exemplo, tio dos nós D e E, ambos filhos de F. A feminista MINI, por seu lado, acha que G é a tia das irmãs H e I, filhas da mãe J. Se você não se importa com o fato de os nós mudarem de sexo em lances alternados, essa analogia o ajudará a compreender melhor a regra alfa-beta:

- Tão logo MAX encontre um filho menos importante que qualquer de seus tios, ignorará os demais irmãos desse filho.
- Tão logo MINI encontre uma filha menos importante que suas tias, ignorará as demais irmãs dessa filha.

No melhor dos casos, o algoritmo alfa-beta examina apenas duas vezes a raiz quadrada do número de nós terminais na árvore do jogo, ao contrário da minimaxização simples. No pior, examina a mesma quantidade de vezes, porém um pouco mais devagar. Para evitar o primeiro dos nossos dois casos, é importante gerar, numa ordem coerente, os irmãos e irmãs a cada nível. Nos níveis maximizantes, devem ser gerados primeiro os melhores nós; nos minimizantes, primeiro os piores (que são os melhores para o oponente).

O jogo de números

Para ilustrar os conceitos mais importantes da pesquisa em árvore, apresentamos um jogo artificial que abrange técnicas de pesquisa quase pura. Isso significa que os detalhes da representação no tabuleiro, geração de movimentos e avaliação estática — fundamentais em qualquer programa de jogo verdadeiro, mas específicos ao jogo em pauta — não mascaram a simplicidade básica do procedimento alfa-beta.

Não há tabuleiro nem peças: o estado do jogo é descrito por um único número, representado por V%. O computador deve elevar o valor de V% para 255; você deve torná-lo menor que 255.

A cada vez, o jogador pode utilizar uma entre quatro funções (A, B, C, D) referidas nas linhas de 1.030 a 1.060 do programa. Você pode

Podando a árvore

Ao eliminar ramificações redundantes da árvore, a "poda" alfa-beta melhora o método básico de minimaxização. Os cortes alfa (indicados por uma barra simples) se formam quando um percurso mínimo já foi encontrado na jogada de MINI, tornando desnecessárias buscas complementares. Os cortes beta (indicados pelas barras duplas) se formam quando é encontrado um caminho máximo com a jogada de MAX.

alterá-las para criar diferentes versões do jogo, tornando-o, por exemplo, mais difícil para o computador.

O jogo, embora simples, ilustra a estratégia da pesquisa, excluindo os detalhes relacionados a um jogo específico como o xadrez. Além disso, é altamente matemático — o computador leva vantagem. Nessa versão, a minimaxização alfa-beta apóia-se no uso de funções recorrentes com parâmetros e variáveis locais. Os parâmetros são os seguintes:

VV%, estado atual do jogo;
A%, melhor valor de alfa até agora nesse nível;
B%, pior valor de beta até agora nesse nível; e
D%, indicador de profundidade.

A versão do programa que fornecemos a seguir serve, praticamente sem alterações, para micros das linhas Apple, TRS-80 e Sinclair.

O jogo dos números

```
70  REM ***JOGO DOS NUMEROS***
80  GOSUB 1000: REM **INICIALIZACAO**
90  GOSUB 1600: REM **INSTRUCOES**
100 :
110 REM ***LOOP PRINCIPAL***
120 GOSUB 2000: REM PREPARAR NOVO JOGO
130 INPUT "QUEM JOGA PRIMEIRO(1=VOCE,2=EU)";H1
140 IF H1 < 1 OR H1 > 2 THEN 130
150 REM ***LOOP DO JOGO***
160 IF H1 = 1 THEN GOSUB 3000
170 REM **VEZ DA PESSOA**
180 GOSUB 3500: REM MOSTRA O TABULEIRO
190 H1 = 1: REM SEMPRE 1 DEPOIS DO PRIMEIRO CICLO
200 GOSUB 4000: REM TESTE DA VITORIA
210 IF EG = 0 THEN GOSUB 5000
220 REM **VEZ DO COMPUTADOR**
230 GOSUB 3500: REM MOSTRA O ESTADO DO JOGO
240 GOSUB 4000: REM TESTE DE FIM DE JOGO
250 IF EG = 0 AND M < = 33 THEN 150: REM VOLTA AO LOOP
260 REM ***FINAL***
270 GOSUB 6000: REM FINALIZACAO
280 INPUT "OUTRO JOGO?(1=SIM,2=NAO)";Y
290 IF Y < 1 OR Y > 2 THEN 280
300 IF Y = 1 THEN 110: REM NOVO JOGO
310 PRINT "ATE LOGO E OBRIGADO PELO JOGO"
320 END
330 :
500 REM **MAXIMIZAR**
510 D = D + 1:C1 = C1 + 1
520 IF D > = MD OR ABS (V(D)) > HI THEN A(D) = V(D):D = D - 1: RETURN

530 REM APROFUNDA NA ÁRVORE
540 P(D) = 0
550 REM LENDO A ARVORE
560 P(D) = P(D) + 1:H = P(D):V = V(D)
565 GOSUB 5500: REM FAZ UMA JOGADA
570 IF D = 1 THEN PRINT CHR$ (64 + H);" = ";
580 D1 = D + 1
590 A(D1) = A(D):B(D1) = B(D):V(D1) = V
595 GOSUB 700: REM CHAMA MINIMIZAR
600 IF B(D + 1) > AD THEN A(D) = B(D + 1):K(D) = P(D)
610 IF D = 1 THEN PRINT B(D + 1);"; ";
620 IF P(D) < = 3 AND A(D) < B(D) THEN 550
625 REM GUARDA O MELHOR ATE AGORA
630 IF D = 1 THEN BV = A(D):HH = K(D)
640 D = D - 1: RETURN
650 :
700 REM **MINIMIZAR**
710 D = D + 1:C2 = C2 + 1
720 IF D > = MD OR ABS (V(D)) > HI THEN B(D) = V(D):D = D - 1: RETURN

730 P(D) = 0
740 REM **ATRAVES DA ARVORE**
750 P(D) = P(D) + 1:H = P(D):V = V(D): GOSUB 5500: REM FAZ UMA JOGADA
760 D1 = D + 1:A(D1) = A(D):B(D1) = B(D):V(D1) = V
770 GOSUB 500: REM CHAMA MAXIMIZAR
780 IF A(D + 1) < B(D) THEN B(D) = A(D + 1)
790 IF P(D) < = 3 AND B(D) > A(D) THEN 740
800 D = D - 1: RETURN
810 :
1000 REM ***INICIALIZA**
1010 CL$ = " "
1020 REM DEFINICAO DAS 4 FUNCOES
1030 DEF FN A(X) = 2 * X - 7
1040 DEF FN B(X) = INT (X / 2) + 1
1050 DEF FN C(X) = 4 * X + 17
1060 DEF FN D(X) = 3 * X - 4
1070 LO = - 255:HI = 255
1080 REM MATRIZES USADAS PELO MINIMAX
1090 D = 10
1100 DIM V(D),A(D),B(D),P(D),K(D)
1110 RETURN
1120 :
1600 REM INSTRUCOES
1610 PRINT "BENVINDO AO JOGO DOS NUMEROS"
1620 PRINT "EU TENTO MAXIMIZAR"
1630 PRINT "VOCE TEM QUE MINIMIZAR"
1640 PRINT "PARA VER O EFEITO DE UMA JOGADA"
1650 PRINT "DIGITE A,B,C OU D E EM SEGUIDA X"
1660 PRINT : RETURN
1670 :
2000 REM **PREPARACAO**
2010 M = 0:V = INT ( RND (1) * 15) - 8: REM ESTADO INICIAL
2020 EG = 0
2030 PRINT "ESTADO INICIAL= ";V
2040 RETURN
2050 :
3000 REM **JOGADA DA PESSOA**
3010 M = M + 1: PRINT
3020 PRINT "SUA JOGADA E ";
3030 INPUT H$
3040 IF H$ = "A" THEN PRINT FN A(V):H = 1
3050 IF H$ = "B" THEN PRINT FN B(V):H = 2
3060 IF H$ = "C" THEN PRINT FN C(V):H = 3
3070 IF H$ = "D" THEN PRINT FN D(V):H = 4
3080 IF H$ < > "X" THEN 3020: REM JOGADA NAO FOI SELECIONADA
3090 GOSUB 5500: REM FAZ UMA JOGADA
3100 RETURN
3110 :
3500 REM MOSTRA O TABULEIRO
3510 PRINT : PRINT "      JOGADA";M;" -->";
3520 IF M < 1 THEN RETURN
3530 PRINT CHR$ (64 + H);
3540 PRINT " = ";V: PRINT : RETURN
3550 :
4000 REM TESTE DE VITORIA
4010 IF M < 1 THEN RETURN
4020 EG = 0
4030 IF V < LO THEN EG = - 1
4040 IF V > HI THEN EG = 1
4050 RETURN
4060 :
5000 REM **JOGADA DO COMPUTADOR**
5010 W = V: REM GUARDA O ESTADO ATUAL
5020 M = M + 1
5030 MD = 6: REM MAX PROFUNDIDADE
5040 IF M < 4 THEN MD = 4
5050 IF M > 8 THEN MD = 8
5060 GOSUB 5200: REM ---> H
5070 V = W: REM VOLTA AO ESTADO INICIAL
5080 GOSUB 5500: REM FAZ UMA JOGADA
5090 RETURN
5100 :
5200 REM SELECIONA A JOGADA
5210 BV = LO:D = 0
5220 V(1) = V:A(1) = LO:B(1) = HI
5230 GOSUB 500: REM MAXIMIZE
5240 H = HH
5245 PRINT
5250 INPUT "TECLE 1 PARA CONTINUAR";G
5255 IF G = 1 THEN 5260
5260 RETURN
5270 :
5500 REM **FAZ UMA JOGADA**
5510 IF H = 1 THEN V = FN A(V): RETURN
5520 IF H = 2 THEN V = FN B(V): RETURN
5530 IF H = 3 THEN V = FN C(V): RETURN
5540 IF H = 4 THEN V = FN D(V): RETURN
5550 :
6000 REM **FINALIZACAO**
6010 PRINT : PRINT "FIM DE JOGO"
6020 IF EG > 0 THEN PRINT "EU GANHEI"
6030 IF EG < 0 THEN PRINT "VOCÊ GANHOU,PARABENS"
6040 IF EG = 0 THEN PRINT "JOGO EMPATADO"
6050 RETURN
6060 :
```

TRABALHO DE BASTIDOR

O que faz do CP/M o sistema operacional baseado em disco mais difundido em todo o mundo? O pioneirismo? A simplicidade e confiabilidade? Vejamos as respostas examinando como ele funciona.

Na primeira metade da década de 70, conquistas sucessivas no campo da informática criaram condições para a revolução dos microcomputadores. Em 1973, surgiu o primeiro microprocessador eficaz de 8 bits — o 8080, da Intel. No ano seguinte apareceu o disco flexível, capaz de agilizar e ampliar consideravelmente os recursos de memória e manipulação de dados computadorizados. Nesse mesmo período, um funcionário da Intel chamado Gary Kindall projetou o CP/M (Control Program/Microprocessors, "programa de controle para microprocessadores"), o primeiro sistema operacional baseado em discos flexíveis destinado a rodar em equipamentos de 8 bits. Em 1975, o CP/M recebia suas primeiras licenças comerciais.

Fazendo trilhas

A maior parte de um disco de dados CP/M está à disposição do usuário, mas as trilhas iniciais são reservadas ao próprio sistema. A trilha zero é a do inicializador, que carrega o CP/M de modo automático e reinicializa o sistema; as trilhas um e dois contêm o próprio CP/M. Seguem-se as trilhas do diretório, que contêm os Blocos de Controle de Arquivo, mostrando onde se localizam os registros de determinado arquivo.

A partir daí, os sistemas operacionais se multiplicaram. Todos, porém, atendem ao mesmo objetivo. Um sistema operacional é um programa em processamento permanente, encarregado de supervisionar as atividades do computador. Incumbe-se do controle de inúmeras tarefas intrínsecas ao funcionamento da máquina, desde formação de uma simples letra no vídeo até o gerenciamento de vários periféricos. Enquanto o sistema operacional atua "nos bastidores", o usuário pode dedicar a totalidade de sua atenção a um programa editor de texto, a uma planilha para cálculos ou a qualquer outro pacote de software que esteja rodando para atender a suas necessidades específicas.

Alguns sistemas operacionais são residentes em ROM — a memória permanente, inacessível ao operador. Por esse motivo, muitos usuários chegam a ignorar sua presença. Costuma-se encontrar esses sistemas fixos em estruturas de computação de menor porte e recursos igualmente fixos, enquanto os chamados DOS (Disk Operating System, "sistema operacional de disco"), carregados na RAM, incrementam a potencialidade do computador em que são executados.

Os DOS apresentam uma diversificação cada vez maior. No entanto, boa parte continua a se inspirar na lógica desenvolvida a partir de 1974 para o CP/M e que o transformou no DOS mais difundido em todo o mundo.

CP/M

Entre os fatores responsáveis pelo êxito do CP/M está sua simplicidade, que permite um acesso fácil e rápido aos arquivos de dados e programas. É, além disso, um sistema adaptável à vasta gama de computadores que utilizam chips 8080 ou compatíveis, o que implica a possibilidade de rodar enorme número de programas.

Quando um computador com CP/M é ligado, uma rotina residente em ROM aciona o Cold Start Loader ("carregador de partida a frio"), um pequeno programa destinado a carregar na memória RAM todo o sistema operacional. Na organização da memória, a área correspondente aos 256 primeiros bytes denomina-se página zero. Trata-se da zona de referência do sistema. Nela estão inscritos dados e indicadores situados em endereços fixos, imutáveis em qualquer sistema ou configuração.

Após a página zero vem a chamada zona TPA (Transient Program Area, "área de programas transitórios"). É a memória utilitária, de dimensões variáveis conforme a configuração do sistema, onde serão carregados os programas e comandos temporários do CP/M. Segue-se o CCP (Console Command Processor, "processador de comandos do terminal"), que analisa e executa os comandos emitidos pelo usuário.

Quando a zona TPA não basta para conter todo um programa em execução, o CCP pode ser "destruído" para criar-se mais espaço. Um RESET (reinicialização) no sistema aciona a rotina

Warm Start ("partida quente"), que acarreta uma expansão da memória TPA.

A parte subseqüente corresponde aos FDOS ("DOS funcional"), subdividido nas zonas BDOS ("DOS básico") e BIOS (Basic Input/Output System, "sistema básico de entrada e saída"). A zona BDOS consiste num conjunto de rotinas especiais para as operações de leitura/registro dos arquivos. Contém o sistema de gerenciamento dos arquivos em disco.

A zona BIOS depende do hardware onde é implantada. Consiste na única porção do CP/M que deve ser adaptada ao computador. Para isso, o fabricante de hardware recebe o programa padrão e reconfigura o trecho final em função do modelo de microcomputador em que o programa será executado.

Tal organização permite distinguir entre comandos residentes (permanentes na memória) e transitórios, carregados no disco e chamados à zona TPA caso necessário. Desse modo, ganha-se capacidade na RAM para as operações de processamento.

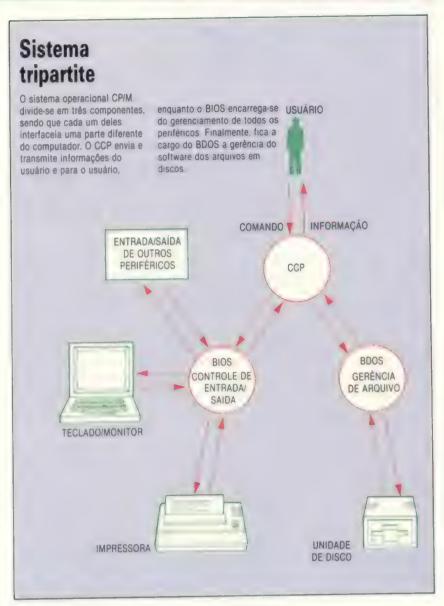
Entre os comandos residentes estão os de consulta ao diretório (índice) do disco, os de eliminação de organização e os de mudança de nome de arquivos. Alguns dos principais comandos relativos a periféricos dizem respeito à comunicação entre estes e os arquivos, bem como à possibilidade de se obterem cópias do próprio sistema operacional.

Apesar de sua confiabilidade e extraordinária popularização, o CP/M revelou, com o tempo, algumas debilidades. Em princípio deveria ser compatível com qualquer tipo de disco, impressora ou terminal, mas a experiência mostrou que não se devem usar discos com capacidade superior a 8 Mbytes.

Outra limitação diz respeito à memória: o CP/M exige pelo menos 20 Kbytes de RAM contínua, o que implica 64 Kbytes para poder operar. Em contrapartida, o TRSDOS, o NEWDOS e outros DOS em que o programa interpretador é residente em ROM exigem apenas 32 Kbytes de memória, embora sejam mais poderosos e mais lentos do que o CP/M.

MP/M

Esse sistema (Multiprogram/Monitor, "multiprograma/monitor") representa um passo adiante em relação ao CP/M a nível de complexidade. Roda todos os programas para CP/M, mas é multiprogramável, podendo trabalhar com até dezesseis terminais (para dezesseis usuários). Os programas não se desenvolvem num lugar fixo na memória, podendo ser colocados em diferentes áreas e executados simultaneamente com outros programas em outras posições. Em outras palavras, o MP/M executa várias tarefas em aparente simultaneidade. De fato, o computador executa seqüencialmente fragmentos mais ou menos entrelaçados dessas diferentes tarefas.

Nesse sistema, cada terminal é um usuário e tem um diretório específico, podendo, entretanto, acessar arquivos do sistema central ou de outros usuários. Mas conseguem-se proteger os arquivos através de senhas.

CP/NET

O CP/NET é um sistema operacional modular para redes, residente num microcomputador principal (em MP/M), em micros secundários (em CP/M) e em vários periféricos de grande porte, velozes e caros.

O desenvolvimento do CP/NET expressa uma preocupação racionalizadora a nível de custos: como utilizar equipamentos caros (os periféricos) sem onerar o preço de cada configuração? A resposta foi a montagem de uma rede, na qual o CP/M e o MP/M não são modificados. Em relação ao requisitante (o CP/NET), a rede controla as operações de entrada e saída, além de enviar mensagens destinadas a periféricos remotos; no servo (o MP/M), a rede consiste num conjunto de processos em andamento e que controla todas as mensagens dos requisitantes, executando as operações necessárias.

RAT

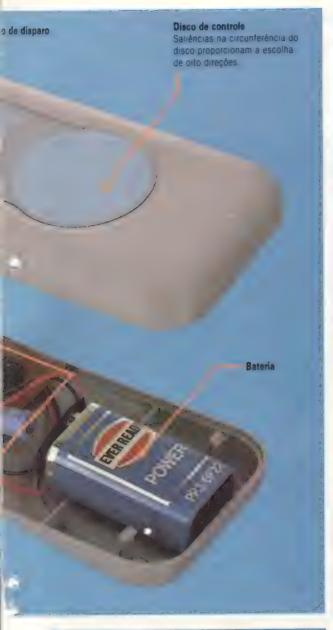
O RAT, um joypad de controle remoto que emprega ondas infravermelhas, é um equipamento fabricado pela Cheetah Marketing. Vamos analisar aqui esse útil periférico para jogos.

Os apreciadores de fliperama costumam se preocupar com qualquer coisa que afete significativamente a velocidade e a qualidade de seus jogos — especialmente se tiver influência notável no escore final. Por essa razão, a forma de controle dos jogos assume extrema importância. Nos dirigidos a partir do teclado, o principal é a escolha das teclas de comando e a facilidade com que podem ser usadas. Em conseqüência, os criadores de software devem dedicar especial atenção a esse detalhe. O design dos controladores externos, como os joysticks, mostra-se um fator crucial.

A flexibilidade de um joystick — liberdade de movimento, rapidez com que reage ao toque e velocidade de resposta do jogo — é a mais significativa consideração em seu desenho. Mas o projetista, com freqüência, se vê tolhido pelas limitações impostas pela natureza do próprio joystick; por exemplo, comprimento do fio de conexão, dimensão, formato e colocação da empunhadura e posição do(s) botão(ões) de disparo. Este último detalhe geralmente desfavorece jogadores canhotos. Embora os fabricantes tentem desenvolver projetos que superem algumas dessas desvantagens, nenhum obteve tanto sucesso como o joypad (tablete para jogos), comando remoto por infravermelho, da Cheetah Marketing, feito para o Sinclair Spectrum.

A Cheetah deu o nome de RAT ("rato") ao seu joypad. Consta que essa sigla seria uma abreviação para Remote Action Transmitter ("transmissor de ação remota"), mas parece ter sido mesmo uma brincadeira com o sinônimo mouse (camundongo), aplicada aos controladores manuais usados com o Macintosh, da Apple, e outros micros. O RAT lembra uma arma phaser, levemente alongada, extraída do seriado de tevê *Jornada nas estrelas*. É longo, achatado, de cor cinza, com um disco azul de controle e um botão de disparo alaranjado. Dois transmissores infravermelhos projetam-se à frente da unidade. Quando se segura o RAT pela primeira vez, quase se espera que ele emita centelhas azuis.

O sistema também inclui sua própria interface, uma caixinha preta que se ajusta no conector edge, na parte posterior do Spectrum. Essa interface tem saída própria para expansões com-

Luz irradiante
A radiação infravermelha é produzida no transmissor por IEDs (Infrared Emitting Diodes, "diodos emissores de infravermelho") quando uma corrente elétrica atravessa um delgado chip de arsenieto de gálio, excitando suas moléculas e liberando fótons. No receptor, inversamente, uma corrente elétrica flui no IED quando a luz infravermelha incide no arsenieto de gálio. Ao se pressionarem os botões de controle do RAT, portanto, os dois IEDs transmissores emitem pulsos codificados de infravermelho em um feixe amplo, disparando o receptor diretamente, ou após a reflexão nas paredes do ambiente.

plementares. Um único receptor infravermelho na frente dessa unidade comunica-se com o RAT.

O pacote contém instruções que explicam como se usa o joypad e em quais jogos se pode aplicá-lo (qualquer software compatível com o joystick Kempston). Com grande visão, a Cheetah incluiu rotinas em BASIC e em código de máquina que possibilitam ao usuário incorporar o controlador RAT a seus próprios jogos.

As instruções afirmam que o RAT pode ser usado a distâncias de até 4 m apenas apontando-se "na direção do computador". O movimento é efetuado pressionando-se levemente o disco azul. Neste, oito pequenas saliências indicam, ao serem pressionadas diretamente ou em suas proximidades, a direção desejada — norte, sudeste, oeste etc., como numa bússola. Enquanto uma das mãos segura o RAT e controla a direção do movimento na tela, a outra comanda o botão de disparo. Devido ao desenho do RAT, não faz diferença qual das mãos realiza cada tarefa, pois o joypad ajusta-se bem tanto para destros como para canhotos. O transmissor requer uma bateria de 9 V, que se encaixa num pequeno alojamento, na parte posterior da unidade, diretamente abaixo do disco azul.

Uma vez conectada a caixa na interface do micro e carregado no Spectrum um jogo que requeira joystick, o usuário acha-se pronto para jogar. Como não há sinal visível de que o transmissor está ou não funcionando até se visualizarem os movimentos na tela, o usuário tende a posicionar-se tão próximo do computador quanto com um joystick comum. Existe certa relutância em aceitar-se uma distância de controle de 4 m; mas, quando se percebe que o RAT realmente funciona, tenta-se experimentá-lo ao máximo "para ver até onde ele vai".

De fato, o transmissor de ação remota funciona muito bem a distâncias até mesmo superiores a 4 m, e não precisa ser apontado em direção ao computador. O RAT atua até quando apontado para o teto, para o chão, para trás do usuário ou para os lados (embora seja um pouco difícil saber o que se está fazendo quando o transmissor é apontado em ângulos estranhos). O aparelho da Cheetah dá ao jogador enorme liberdade de movimentação. A maior desvantagem, entretanto, reside nas oito posições de movimento — acima, abaixo, direita, esquerda e os quatro pontos intermediários. Seria melhor se tivesse mais opções de controle.

O fato de não ter partes móveis torna o RAT uma unidade menos propensa a desgaste e ruptura do que os joysticks comuns, devendo ter uma vida útil bastante longa. De fato, a unidade vem com garantia de um ano. O transmissor da Cheetah custa somente um pouco mais que a maioria dos outros joysticks acrescidos da Interface 2 (o que não consola muito, se você já tem a Interface 2). Mas sua flexibilidade permite muito mais liberdade de movimento e um controle muito superior ao da maioria dos joysticks.

Recepção mista
O teclado original do fracassado PC Junior, da IBM, notabilizou-se principalmente pela pobreza de aspecto e configuração; mas esse micro teve o mérito de ser o primeiro com ligação infravermelha entre o teclado e o processador.

TELEDADOS

ONDE ESTUDAR
UNI
1 JARDIM/PRÉ-ESCOLAS ?
2 1° GRAU
3 2° GRAU
4 FACULDADES
Voltar: ED + envio Tecle n° + envio

O mundo na tela
O videotexto coloca ao alcance do usuário bancos de dados com informações de praticamente todas as áreas do conhecimento humano. Até mesmo do exterior. Digitando palavras-chaves, o assinante vai selecionando os registros até chegar ao serviço que deseja. As informações podem ser, então, lidas, gravadas em disquete, em cassete ou impressas em papel.

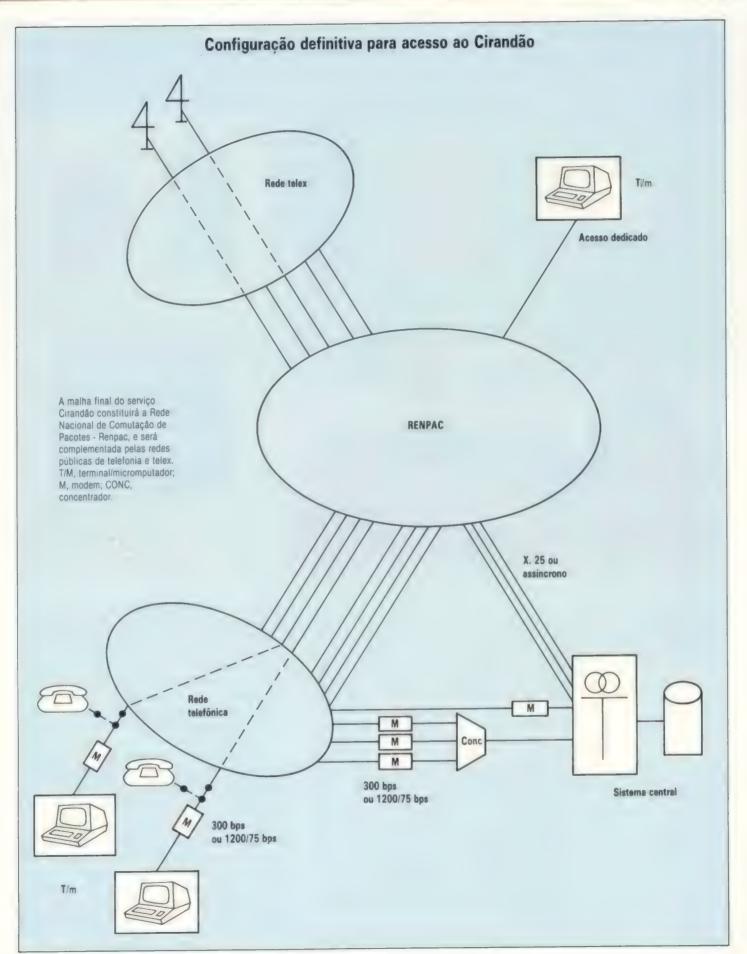
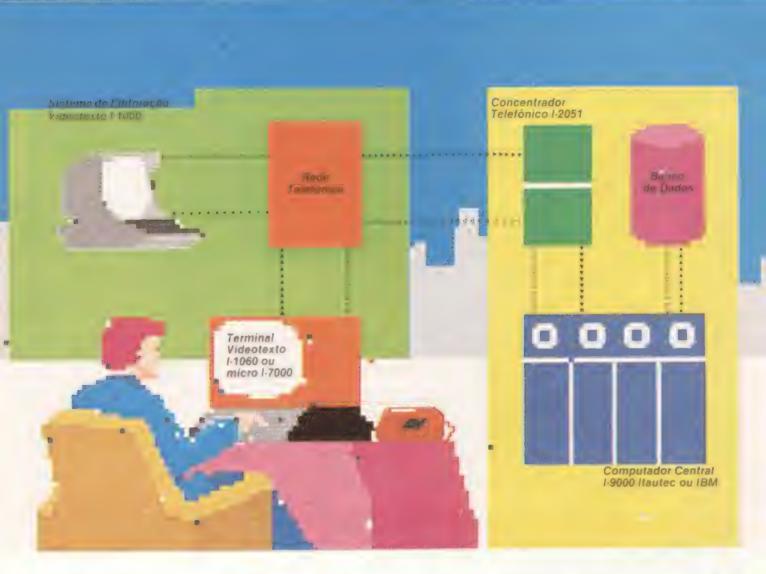
Talvez nenhum outro recurso tenha, como o videotexto, simplificado de modo tão acentuado o esforço pela busca da informação. O tempo se reduz a segundos e o espaço, à distância entre dedo e tecla.

O videotexto nasceu da aliança de dois dos meios de comunicação mais poderosos, o telefone e a televisão. Por meio de um terminal, acessam-se todas as informações não reservadas contidas num banco de dados de uma empresa pública ou privada, que são exibidas numa tela de televisão ou de microcomputador.

O fluxo de bytes que contém a informação desejada é convertido em impulsos elétricos, que percorrem a linha telefônica. Estes, atingindo o modem (conversor), a uma velocidade de cerca de 1.200 bauds (bytes por segundo), são reconvertidos em bytes. O modem reenvia então o fluxo de bytes ao terminal de videotexto, que o transforma em palavras e números exibidos no vídeo. Selecionam-se as informações desejadas pela digitação de palavras-chaves no teclado. Se possui uma impressora, o usuário pode obter uma cópia da informação recebida, ou então gravá-la em disquete ou fita cassete.

O videotexto inaugurou um caminho extremamente rápido de acesso a informações, tornando supérfluos tanto o acúmulo de papel quanto

Configuração definitiva para acesso ao Cirandão

A malha final do serviço Cirandão constituirá a Rede Nacional de Comutação de Pacotes - Renpac, e será complementada pelas redes públicas de telefonia e telex. T/M, terminal/micromputador; M, modem; CONC, concentrador.

O sistema em uso
Esquema ilustrando como funciona o equipamento oferecido pela Itautec para serviços de edição de textos.

a lentidão associada às vias tradicionais de acesso à documentação escrita. Além disso, reduz bastante os prazos de várias formas de transação envolvendo papel, bem como os custos decorrentes dos métodos tradicionais para produzir impressos. Além disso, novas informações recebidas por um banco de dados são acrescentadas às já existentes ou substituem as mais antigas, mantendo o acervo sempre atualizado.

A quantidade de informações a que já se pode ter acesso, no Brasil, através dos atuais bancos de dados, chega a ser assombrosa. Não é apenas a extensão dessa rede que impressiona, mas também sua profundidade: em geral, acessam-se os registros, por meio de uma estrutura em árvore; ou seja, a cada opção feita, surgem novas opções, cada vez mais específicas. Diz-se que o usuário "navega" na árvore, seguindo o aprofundamento progressivo da informação. Navega-se por intermédio de comandos que manipulam menus e selecionam itens, números de páginas, palavras-chaves.

Espera-se que o progresso do videotexto seja acompanhado por uma correspondente evolução das formas de busca. Assim, a navegação pela árvore alia-se a elementos visuais no videotexto de segunda geração, cujos protótipos estão sendo estudados. Nele, o acesso às opções desejadas é feito através da imagem, assimilada a "caixas" que vão sendo sucessivamente abertas. Esse processo elimina a mediação do teclado. As caixas se abrem por meio de pontos na figura, ativados por periféricos como o joystick, o mouse ou a caneta óptica.

A Itautec, líder no mercado nacional de videotexto, oferece um equipamento completo de hardware e de software para gerenciar o sistema de editoração, abrigado em disquetes de 5 1/4 polegadas. O terminal para videotexto Itautec VDTX I-1060 compõe-se de adaptador, teclado, monitor ou televisor (opcionais) e cabos de conexão a gravadores. O primeiro manipula toda a eletrônica necessária para o uso do terminal, através de um módulo de controle, um microprocessador Z80, uma saída VHF (que permite ligar o terminal a um aparelho de tevê), uma saída RGB para vídeo, duas para gravador cassete e uma para fonte de alimentação.

O teclado, padrão QWERTY, possui 44 teclas alfanuméricas, quinze de função e duas para futuras expansões. Além do terminal, a Itautec possui o sistema de editoração de videotexto I-1000, com o qual o fornecedor de serviços pode estruturar os programas de videotexto, criando, armazenando e atualizando as páginas exibidas no vídeo. A placa de interface para I-7072 contém todo o hardware necessário para transformar o micro I-7000 da Itautec num terminal para videotexto. É também parte integrante do sistema de editoração. Dois outros equipamentos, o concentrador telefônico I-2051 e a unidade de controle de comunicação UCCI I-4010, permitem a comunicação com terminais remotos.

No Brasil, alguns dos bancos de dados mais importantes são o do sistema Cirandão, da Embratel (que também oferece o sistema Interdata), o Aruanda, da Serpro, e o Videotexto, da Telesp. O procedimento de acesso ao Cirandão ilustra bem a facilidade de manipulação do equipamento. Discando um número apropriado de telefone, o usuário ouve um sinal contínuo. Transfere, então, a ligação para o modem. Após 5 segundos, digitará BREAK (ou NULL) para o sistema central, o qual, feita a conexão, pedirá ao usuário para se identificar através de uma senha. Pressionada a tecla [Enter], aparecerá o menu principal. Digitando [C] (Comunicações) nesse menu e [M] (de Mensagens Pessoais) no segundo, tem-se acesso à caixa postal, que permite receber ou enviar mensagens a outros usuários do Cirandão.

A importância dos serviços prestados pelo videotexto fica patente quando se pensa num banco de dados como o Cirandão Agropecuário, que fornece informações importantes sobre cultura e criação, instruções ao combate de pragas, preços de produtos e equipamentos, previsão do tempo etc., registros enviados por vinte instituições, além de vários centros de pesquisa. Outro importante banco de dados do sistema é o Cirandão Saúde, bem equipado em informações de várias organizações médicas e farmacológicas.

Já não há, no Brasil, praticamente nenhuma área de interesse público — ou mesmo restrito a setores mais especializados — desprovida da possibilidade de acesso a banco de dados. Assim, a Niterói Comércio Exterior e Despachos

Novidades à mão
O assinante do serviço de videotexto tem a seu alcance grande variedade de jogos e brincadeiras. O sistema permite acessá-los em segundos, carregá-los na memória do micro ou gravá-los.

Facilidade de impressão
O I-1000 para editoração de textos, da Itautec. Cada página é digitada no teclado e editada no vídeo. Em seguida, é gravada em disquete e posteriormente armazenada no computador central. Está pronta para ser acessada pelo usuário. O editor de textos vem se mostrando uma profissão em ascensão no mercado de trabalho da informática.

Aduaneiros Ltda. informa as datas de entrada e saída de navios, legislação do comércio exterior, custos sobre exportações e importações etc. A Prodam reúne informações sobre a CMTC, a Prefeitura de São Paulo, trânsito, habitação, multas e imposto predial (a maioria dessas informações, no entanto, é restrita a empresas de trânsito e secretarias de Estado). A Embravídeo oferece ao público em geral um banco de memória de comerciais brasileiros e estrangeiros, de consulta gratuita. A Bovespa fornece dados sobre o mercado de ações em vários Estados brasileiros. O banco de programas do Cirandão tem interesse para várias áreas administrativas, além de jogos e informações científicas e educacionais.

O serviço Interdata, da Embratel, permite o acesso a bancos de dados localizados no exterior. Um desses é o Dialog, que reúne informações de mais de 150 bases, com informações científicas, tecnológicas e econômicas, entre outras, como a história da América do Norte, educação, patentes e direitos autorais, e química. A Questel Telesystèmes, banco de dados francês, reúne 35 bases, como a Cancern, que informa sobre assuntos como bioquímica e imunologia, e a Cécile, onde se podem obter registros sobre desenho industrial, comunicação visual, arquitetura. A Dow Jones, que, como a Questel, pode ser acessada via Embratel, fornece resumos das notícias financeiras do *Wall Street Journal*.

A Telesp oferece vários serviços de interesse mais geral: roteiros turísticos, noticiário nacional e internacional, reservas de passagens em vôos nacionais, extratos de contas correntes do Bradesco e o serviço Teleshopping, que possibilita a compra de eletrodomésticos, brinquedos, artigos de cama e mesa etc. A utilização dos serviços da Telesp é cobrada na conta telefônica do usuário. O Museu Cultural França-Brasil e o SID (Sistema de Documentação e Informação), do Instituto Goethe, foram duas importantes adesões. O banco de dados do primeiro registra detalhes do relacionamento Brasil-França desde a época do Descobrimento. Um outro terminal comunica-se, via satélite, com o Centro Pompidou, em Paris, ligado a uma rede de cerca de setenta bancos de dados, com acesso a informações fornecidas por universidades, centros de pesquisa e bibliotecas.

Aliás, a França, um dos países pioneiros na implantação de videotexto, já conta com mais de 2 milhões de terminais instalados, e a previsão é de 11 milhões até o fim da década. O SID do Instituto Goethe comunica-se on line com o banco de dados central da Alemanha; este, através do sistema INKA, troca informações com outros bancos do mundo inteiro. Assim, tem-se acesso a 15 milhões de itens, volume que, a cada ano, aumenta em 500.000 novas informações científicas e culturais.

O videotexto pode emprestar seus próprios recursos à palavra escrita, abrindo novos caminhos à literatura. Interessante exemplo disso é o multiconto, da Telesp, criação do escritor Renato Pompeu; conto único que se desdobra em 160 histórias, cabendo ao usuário decidir sobre o destino que quer dar a cada personagem. Diferentes opções vão montando histórias diferentes. O videotexto introduz, assim, a decisão como um fator novo no processo de leitura e, naturalmente, um novo tipo de envolvimento do leitor com o texto.

O serviço também oferece seus recursos à arte da programação gráfica. As férteis possibilidades da manipulação artística de imagens de baixa definição podem ser evidenciadas nos painéis digitais. São figuras cuja metamorfose e diluição têm tanto peso quanto elas próprias. Artistas como Júlio Plaza e poetas como Décio Pignatari são pioneiros em explorar a arte desse "videotexto de imagens". É provável que um desenvolvimento, nesse novo meio de comunicação, de horários "reservados a comerciais", promova muito essa área, ainda tímida do videotexto; tanto como poderá também promover e afetar os próprios rumos do videotexto. A importância desse novo meio na área educativa é outro motivo que certamente fará da arte da programação gráfica um poderoso auxiliar.

A esses fatos alia-se a ênfase que o videotexto de segunda geração pretende dar à programação gráfica. As perspectivas para o futuro são ainda mais fascinantes. A alta resolução da imagem por pixels revela-se um campo no qual a pesquisa caminha a passos largos. Imagens digitais de alta definição podem ser ampliadas localmente — aliás, uma das características mais curiosas da televisão digital. Hoje, o custo para tal implantação seria impraticável, mas é possível que o videotexto do futuro comporte bancos de dados fotográficos. O usuário teria então acesso ao acervo dos museus de arte usufruindo as pinturas por meio de ampliações locais.

RASCUNHO ELETRÔNICO

A pilha é uma área de trabalho que permite, através de instruções próprias, copiar e restaurar o conteúdo dos registradores. Examinamos aqui a pilha e suas operações, essenciais na execução de sub-rotinas em ASSEMBLY.

A essência da linguagem ASSEMBLY é a manipulação da memória; daí porque as instruções até aqui estudadas servirem, na maioria, basicamente para carregar dados em determinadas posições de memória, ou transferi-los para outros locais. A maneira de acessar tais posições varia conforme o modo de endereçamento, mas o endereço sempre faz parte do operando. Existem, entretanto, certas instruções que acessam uma área da memória, mas não usam o endereço como operando. Elas operam na área de memória conhecida como pilha, motivo pelo qual se chamam operações de pilha.

A pilha — uma área temporária de trabalho, disponível tanto para o programador como para a CPU — é um tipo de bloco de rascunho, onde facilmente se lêem, gravam e apagam informações. As operações de pilha copiam dados dos registradores nas áreas desocupadas da pilha, ou desta naqueles. Tais instruções não requerem endereços como operandos, pois há um registrador específico para esse fim — o ponteiro da pilha —, que sempre contém o endereço da próxima posição livre na pilha. Assim, qualquer comando para gravar dados na pilha irá utilizar o byte indicado pelo ponteiro, e os comandos que copiam dados da pilha irão acessá-los no endereço onde foram gravados dados pela última vez. A execução de qualquer operação de pilha já inclui o reajuste do ponteiro.

Nos equipamentos baseados no processador 6502, a pilha ocupa 256 bytes de RAM, da posição $0100 até $01FF; no Z80, a localização e o tamanho da pilha são determinados pelo sistema operacional, e modificáveis pelo programador. Essa variação reflete as diferenças na organização interna dos dois microprocessadores: o ponteiro de pilha do 6502 tem um único byte, enquanto o do Z80 tem 2 bytes.

No 6502, a CPU considera o conteúdo do ponteiro como o byte inferior do endereço da pilha; a ele se acrescenta automaticamente um byte superior de $01, por meio de um "nono bit" associado ao ponteiro. Esse bit extra sempre tem o valor 1 e, assim, os endereços de pilha no 6502 ficam sempre na página 1.

O ponteiro de pilha do Z80 é um registrador de 2 bytes, capaz de indicar qualquer localização entre $0000 e $FFFF — ou seja, todo o espaço endereçável do próprio processador Z80. Assim, a pilha pode situar-se em qualquer lugar da RAM, sendo sua posição alterável pelo programador. Mas não se recomenda tal procedimento, pois o sistema operacional inicializa a localização da pilha e nela armazena dados. O sistema operacional pode interromper a qualquer momento a execução de um programa em código de máquina e, nesse caso, irá procurar na pilha os dados necessários para sua operação; assim, qualquer mudança na localização da pilha implicará a ausência dos dados relevantes, resultando até num colapso do sistema.

Para exemplificar o uso da pilha, veja a rotina abaixo, que troca o conteúdo de dois locais de memória, LOC1 e LOC2. As instruções PUSH e POP do Z80 indicam, respectivamente, a ação de colocar e a de retirar dados da pilha.

6502		Z80	
LDA	LOC1	LD	A,(LOC1)
PHA		PUSH	AF
LDA	LOC2	LD	A,(LOC2)
STA	LOC1	LD	(LOC1),A
PLA		POP	AF
STA	LOC2	LD	(LOC2),A

Inicialmente se carrega o conteúdo de LOC1 no acumulador; deste, eles são copiados para a pilha. Em seguida, carrega-se no acumulador o conteúdo de LOC2, que daí é armazenado em LOC1. Copia-se então o conteúdo do byte superior da pilha para o acumulador, que passa assim a armazenar o conteúdo original de LOC1. Transfere-se então este para LOC2, completando-se a troca. Note que as operações de pilha "salvaram" o conteúdo de LOC1 na memória pelo tempo necessário, mas o programa não especificou nenhuma posição para esse fim. Por implicação, essa posição foi suposta como a próxima localização livre da pilha.

Nosso programa-exemplo esclarece vários outros pontos sobre as operações de pilha. Basicamente, elas são recíprocas e seqüenciais. O último item colocado na pilha é recuperado pela próxima operação de extração. Se houver várias introduções de dados sem nenhuma extração, os dados irão sendo gravados em posições sucessivas da pilha, um "em cima" do outro; da mesma forma, havendo várias extrações de dados sem qualquer acréscimo, os dados irão sendo acessados "de cima para baixo" na pilha.

Para visualizar melhor esse processo, imagine-se escrevendo vários cartões postais e empilhando-os à sua frente; ao terminar, irá ler um por um até acabar a pilha, antes de enviá-los. O último cartão escrito estará no topo da pilha. Esse tipo de estruturação dos dados chama-se LIFO (Last In First Out, "último a entrar, primeiro a sair"). O inverso — a FIFO (First In First Out, "primeiro a entrar, primeiro

a sair") — é uma simples fila, como as que enfrentamos no dia-a-dia. Apenas por convenção considera-se que a pilha cresce para cima e que o próximo byte livre seja o do topo. Na verdade, tanto no Z80 como no 6502, cada introdução de dados irá decrementar o ponteiro, de forma que o topo da pilha tem um endereço de memória inferior ao da base. Podemos dizer que a pilha "cresce em direção a zero".

Também observamos na rotina de exemplo que o número de instruções que introduzem dados é idêntico ao das que extraem dados. Normalmente se trabalha com pilhas dessa maneira; não é essencial, mas, se desrespeitarmos esse equilíbrio entre os opostos, as sub-rotinas que escrevemos poderão ter um retorno incorreto, ocasionando falha no programa. Esse é um dos erros mais comuns nos programas em ASSEMBLY, mas de fácil correção: basta comparar no programa o número de instruções de introdução com o de extração de dados.

Há grande diferença entre a versão Z80 e a 6502: esta sempre coloca na pilha registradores de um só byte; aquela, de 2 bytes. No Z80, ao se introduzir ou extrair o conteúdo do acumulador, também se introduz ou se extrai o conteúdo do registrador de estado de processador. Isso porque a CPU trata esses dois registradores de um só byte como um só registrador de 2 bytes, chamado registrador AF (Accumulator Flag, "flag do acumulador"). A potência do Z80 deriva em grande parte de sua capacidade de manipular registradores de 2 bytes.

A boa técnica de programação recomenda, ao se iniciar uma sub-rotina, que se coloque na pilha o conteúdo de todos os registradores da CPU e, logo antes de se retornar dela, que se retire esse conteúdo. Isso vai assegurar que a CPU, após a execução da sub-rotina, continue num estado idêntico ao anterior. Significa também que se pode utilizar na sub-rotina qualquer dos registradores sem risco de adulterar dados essenciais ao programa. Por exemplo, considere esta sub-rotina de um programa:

	6502		Z80	
	LDA	LOC1	LD	A,LOC1
SUM	ADC	#$6C	ADC	A,$6C
GSUB	JSR	SUBR0	CALL	SUBR0
TEST	BNE	SUM	JR	NZ,SUM
EXIT	RTS		RET	
SUBR0	PHP		PUSH	AF
	PHA		PUSH	HL
	TXA		PUSH	DE
	PHA		PUSH	BC
	TYA		PUSH	IX
SUBR1	PHA		PUSH	IY
SUBR2	STA	LOC2	LD	(LOC2),A
	LDA	#$00	LD	A,$00
SUBR3	PLA		POP	IY
	TAY		POP	IX
	PLA		POP	DE
	TAX		POP	BC
	PLA		POP	HL
SUBR4	PLP		POP	AF
	RTS		RET	

ANTES

CONTADOR DO PROGRAMA

E946

$0100
$015F PRÓXIMO BYTE LIVRE
$01FF

PILHA

PONTEIRO DA PILHA

INSTRUÇÃO

JSR $F000

JSR $E943
00 $E944
F0 $E945
CLC $E946
RTS $F000

MEMÓRIA DO PROGRAMA

DEPOIS

CONTADOR DO PROGRAMA

F000

$0100
$015D PRÓXIMO BYTE LIVRE
$015E 46
$015F E9
$01FF

PONTEIRO DA PILHA

PILHA

A pilha

A pilha é essencial nas chamadas de sub-rotinas (como a instrução JSR do 6502). Quando se carrega pela primeira vez a instrução "saltar para a sub-rotina", o contador do programa contém o endereço da próxima instrução (aqui, $E946, o endereço da instrução CLC), este, colocado "no topo" da pilha, faz com que o ponteiro passe a apontar o próximo byte livre desta (aqui, $015D). O operando da instrução de salto (aqui, $F000) é carregado no contador do programa como endereço da próxima instrução, causando assim um desvio no fluxo do programa.

As instruções entre SUBR0 e SUBR1 colocam na pilha o conteúdo atual do registrador; as instruções entre SUBR3 e SUBR4 restauram nos registradores esse conteúdo. As instruções essenciais são as duas que iniciam em SUBR2, mas a segunda delas não tem efeito, uma vez que as subseqüentes alteram por completo o estado do acumulador.

Note que as instruções PUSH e POP do Z80 tomam como operando qualquer dos pares de registradores; o 6502, contudo, trabalha apenas com o acumulador (PHA e PLA) e com o registrador de estado do processador (PHP e PLP). Daí vem a necessidade, na versão 6502, de transferências entre registrador e acumulador (TXA, TAX, TYA, TAY). Note também que, na versão Z80, cometemos deliberadamente um erro ao não extrair todos os registradores na ordem inversa em que foram colocados na pilha. Isso ilustra o cuidado que se deve tomar em operações de pilha, e demonstra que é possível introduzir dados a partir de um registrador e extraí-los de volta para outro — uma maneira trabalhosa, mas às vezes conveniente, de transferir dados entre registradores.

No próximo artigo examinaremos os usos e funções dos registradores da CPU, e iniciaremos o estudo da aritmética do código de máquina.

Exercícios

A) Reescreva a segunda rotina dada nas "Respostas aos exercícios anteriores", de modo que a mensagem em LABL1 seja armazenada de novo no mesmo local, mas na ordem inversa, assim:

LABL1 MEGASNEM AMU SIE

Use a pilha para essa inversão.

B) Desenvolva essa rotina para que as palavras da mensagem permaneçam em sua ordem original, mas com os caracteres de cada palavra invertidos, assim:

LABL1 SIE AMU MEGASNEM

Respostas dos exercícios anteriores

A) Esta sub-rotina armazena os números de $0F a $00, em ordem decrescente, no bloco de $10 bytes reservado pelo pseudocódigo DS em LABL1.

6502		Z80	
ORIGIN	ORG $7000	ORIGIN	ORG $C000
LABL1	DS $10	LABL1	DS $10
LABL2	DW $7100	LABL2	DW $C100
		OFFST	EQU $0F
BEGIN	LDY #$FF	BEGIN	LD IX,LABL1
	LDX #$10		LD B,OFFST
LOOP0	INY	LOOP0	LD (IX+0),B
	DEX		INC IX
	TXA	ENDLP0	DJNZ LOOP0
	STA LABL1,Y		LD (IX+0),B
ENDLP0	BNE LOOP0		RET
	RTS		

O 6502 usa o registrador Y como índice para o endereço LABL1 e o registrador X como contador do loop e fonte dos dados a serem armazenados. Note que o registrador X é decrementado duas instruções antes do teste BNE em ENDLP0. Entretanto, como STA e TXA (tranferir o conteúdo de X para o acumulador) não afetam o registrador de estado do processador, o teste verifica o efeito da redução de X.

A versão Z80 usa o modo de endereçamento indireto IX para manter o endereço de armazenamento e o registrador B como contador e fonte de dados. Em ENDLP0, vemos DJNZ LOOP0, que significa "decrementar o registrador B e dar um salto relativo para LOOP0 se o resultado for diferente de zero". Essa instrução quase corresponde, em ASSEMBLY, à estrutura FOR-NEXT, e certamente torna mais fácil e conveniente escrever loops no Z80.

B) Essa rotina copia a mensagem armazenada em LABL1 para o bloco que inicia no endereço armazenado em LABL2. Armazena-se no fim da mensagem, como indicador de término, o valor $0D (o código ASCII para Return ou Enter).

6502		Z80	
ORIGIN	ORG $7000	ORIGIN	ORG $C000
LABL1	DB 'EIS UMA MENSAGEM'	LABL1	DB 'EIS UMA MENSAGEM'
TERMN8	DB $0D	TERMN8	DB $0D
LABL2	DW $7100	LABL2	DW $C100
CR	EQU $0D	CR	EQU $0D
ZPLO	EQU $FB		
BEGIN	LDA LABL2	BEGIN	LD IX,LABL1
	STA ZPLO		LD IY,(LABL2)
	LDA LABL2+1	LOOP0	LD A,(IX+0)
	STA ZPLO+1		LD (IY+0),A
	LDY $FF		INC IX
LOOP0	INY		INC IY
	LDA LABL1,Y		CP CR
	STA (ZPLO),Y	ENDLP0	JR NZ,LOOP0
	CMP CR		RET
ENDLP0	BNE LOOP0		
	RTS		

A versão 6502 usa o registrador Y como índice para o endereço indireto ZPL0 no modo de endereçamento pós-indexado. Esse modo é possível apenas com o registrador Y e requer um endereço de operando de página 0 — daí a inicialização de ZPL0 e ZPL + 1 com o endereço armazenado em LABL2. O sistema operacional das máquinas baseadas no 6502 usa a maioria das posições da página 0. A versão Z80 usa IX no modo indexado e IY no modo indexado indireto.

Ambas as rotinas usam uma instrução "comparar o acumulador" — CMP CR (6502) e CP CR (Z80) — na qual se subtrai o operando do conteúdo do acumulador, afetando assim os flags do registrador de estado do processador (PSR). Restaura-se então o conteúdo do acumulador, enquanto o PSR mostra os resultados da comparação. Quando o acumulador contém $0D (o indicador de término de mensagem), o resultado da comparação é a ativação do flag 0. Assim, o teste ENDLP0 falhará, passando o controle à instrução de retorno.

MICROSOFT

A Microsoft tornou-se um dos maiores fornecedores de software para microcomputadores, e participou da formulação das especificações do IBM PC, o mais vendido do mundo.

A Microsoft, hoje uma empresa milionária, é uma história clássica de entusiasmo bem-sucedido. Bill Gates, de 28 anos, seu presidente, em 1972 era apenas amador talentoso.

Na Seattle High School — que a associação de pais e mestres teve a feliz idéia de equipar com um terminal de computador ligado a um míni tipo DEC PDP-11 —, Bill aprendeu as noções básicas de informática. Estudou depois na Universidade de Harvard, onde obteve sua graduação. Ao voltar a Bellevue, iniciou-se nos negócios com o colega de universidade Paul Allen.

A empresa que então fundaram chamava-se Traff-D-Data, cujo trabalho consistia em monitorar o fluxo do tráfego para o departamento de trânsito de Seattle. Ocorria um período de desenvolvimento extraordinário do microcomputador: os primeiros microprocessadores estavam surgindo e quem possuía um pouco de visão e entusiasmo percebeu o grande futuro dos chips, como o Intel 4004 e mais tarde o 8008. Na ocasião, Bill já estava inteiramente familiarizado com o míni DEC PDP-11, e um de seus primeiros objetivos foi tentar localizar deficiências nesse equipamento. Ocorreu-lhe que seria uma boa idéia adaptar o BASIC desse míni para uso com o chip. Ele não tinha um sistema de desenvolvimento, e a primeira ocasião em que o BASIC rodou na máquina foi quando Gates levou as fitas para Altair, em Albuquerque, Novo México. Incrivelmente, rodou sem problemas logo da primeira vez. Assim nasceu a linguagem MBASIC, que passou a ser empregada como padrão, até agora insuperado.

A Microsoft estava ficando conhecida como uma software house especializada na adaptação de sistemas operacionais a novos computadores — preencher buracos, por assim dizer. Foi quando a IBM entrou em contato com Gates, solicitando seu conselho sobre como configurar e equipar um computador pessoal.

Inicialmente Gates sugeriu que Gary Kildall, da Digital Research — então no auge da fama, com o florescente sucesso do sistema operacional CP/M —, seria o homem indicado para o serviço. Mas a IBM voltou a procurá-lo, e a Microsoft acabou reescrevendo as linguagens PASCAL, FORTRAN e MBASIC para micros de 16 bits, além de criar a linguagem GU (de "geewhizz", uma exclamação de surpresa) BASIC, com capacidades musicais e gráficas ampliadas.

Ao mesmo tempo, Gates percebeu que um sistema operacional multiusuário — um tanto desorganizado, mas potente — da Bell Laboratories poderia ser adaptado para os micros mais poderosos baseados nos novos microprocessadores de 16/32 bits. E transformou o Unix no Xenix, que a Tandy e a Apple adotaram em seus próprios modelos de 16/32 bits, em 1983. Chegou a transpirar que a Microsoft teria sido responsável por grande parte da nova criação da Apple, o microcomputador Macintosh.

A Microsoft tem participação apreciável também no mercado amador. Em 1981, ela fundou a ASCII-Microsoft e colocou um talentoso jovem japonês, Kay Nishu, para vender seus sistemas operacionais e o BASIC aos fabricantes de micros portáteis do Extremo Oriente — por exemplo, o NEC PC 8201 e o Tandy Model 100. Como resultado do desejo dos fabricantes japoneses de características comuns, não somente para as linguagens, mas também nas interfaces para periféricos, desenvolveu-se o padrão MSX. O próximo passo da Microsoft, ao que tudo indica, será o formato padronizado para discos, que possibilitará a transferência de informações entre os três principais ambientes operacionais — o MSX, o MS-DOS e o Xenix.

Com sua ênfase em softwares fáceis de usar, e baseados em recursos avançados como as janelas na tela e o mouse, a Microsoft parece ter pela frente um brilhante futuro.

[illegible]

Em 1970, com a idade de 28 anos, Shiina Takayoshi abandonou uma promissora carreira militar e fundou a Sord Corporation. Ele imediatamente formulou onze princípios para dirigir sua nova empresa de computação. Alguns desses princípios eram os seguintes:

- A principal obrigação da companhia é para com a humanidade.
- A companhia deve esforçar-se ao máximo a fim de determinar quais os melhores produtos e serviços para a sociedade, e fornecê-los a um custo razoável.
- Não haverá qualquer divisão entre os funcionários e a administração. Todas as pessoas da companhia devem se respeitar mutuamente e cooperar para o benefício de todos.

Padrão da indústria

A linguagem BASIC — (Beginners All-purpose Symbolic Instruction Code, "código de instrução simbólica para todos os fins, para principiantes") — foi desenvolvida em 1965 no Dartmouth College, EUA por J. Kemeny e T. Kurtz, antecede, pois, o microprocessador em pelo menos sete anos. Embora muitas versões dessa linguagem tenham sido criadas, o MBASIC (a versão da Microsoft) acabou reconhecido como o padrão industrial.

A Microsoft firmou sua reputação no mercado em cima do sucesso do MBASIC e continuou a prosperar, produzindo um sério desafiante ao CP/M — o MS-DOS, um sistema operacional projetado para aplicação numa ampla gama de micros.

Seguindo a liderança determinada pelo sistema terminal Star, da Xerox — desenvolvido pela Apple com o Lisa e o Macintosh —, a Microsoft produziu um pacote combinando o software que divide a tela em janelas independentes (o MS-WINDOWS) e o dispositivo para manejá-las: um mouse com uma esfera conjugada a dois seletores para deslocar o cursor pela tela.

GUERRA DE IMAGENS

As aparências contam
Estes dois programas custam aproximadamente a mesma coisa. Um deles vem numa embalagem criativa, que atrai a atenção e sugere um bom investimento, enquanto o outro, comparativamente, dá impressão de uma caixa feia e bem pouco interessante por um preço alto.

Seja vendendo o último videojogo lançado ou o mais recente computador profissional apoiado por considerável verba publicitária, o fator fundamental do sucesso continua sendo a criação da imagem certa para o produto.

Marketing é a arte de construir uma ponte entre o produto que se tem para vender e o bolso dos possíveis compradores. Ao planejarem tal ponte que incentive e dê vazão máxima às mercadorias, os agentes de venda arriscam "palpites" (às vezes com base em pesquisas de mercado) sobre as necessidades, desejos e caprichos do público-alvo, e então sustentam esses palpites com grandes investimentos. Por exemplo, o lançamento do Macintosh custou, no seu auge, só na Inglaterra, 1 milhão de libras por mês.

A estratégia de vendas começa a ser decidida já no estágio de planejamento de uma nova máquina. Quais as funções do equipamento, quantas unidades serão produzidas, quanto deverá ser gasto na fabricação de cada unidade etc. constituem fatores que pesam ao se traçarem as linhas da comercialização.

Considerações semelhantes aplicam-se ao marketing de softwares. De nada adiantará gastarmos somas enormes no desenvolvimento e veiculação de um videojogo que irá custar mais do que seus possíveis compradores poderão desembolsar — se quisermos que ele pague o investimento. Em contrapartida, uma empresa de software que limita o investimento em pesquisa poderá se ver às voltas com a comercialização de um produto pobre em recursos, cheio de erros, ou ambas as coisas. Corre então o risco de assistir sua marca comercial cair no descrédito, com efeitos desastrosos para os produtos futuros. Se, por outro lado, um equipamento receber toda a atenção no desenvolvimento, mas pouca ou nenhuma na comercialização, resultará um magnífico produto sobre o qual ninguém ouviu falar.

Tem muita importância logo de início onde o equipamento se encaixa em relação aos concorrentes. É aqui que a imagem do produto — um tipo de projeção psicológica que sintetiza o objeto — se torna fundamental. Os fabricantes de cigarros e de sabão em pó partilham de uma "pequena" dificuldade: se se disser a verdade, cada produto será igual aos concorrentes. Os fabricantes de hardware para computadores, entretanto, não se encontram nessa incômoda posição, embora nem sempre seja fácil explicar com precisão as particularidades de um equipamen-

Imagens
Nosso exemplo pode facilmente mostrar a importância de se criar uma imagem poderosa do produto no mercado: o elefante da Commodore sugere a memória "jumbo" do Commodore 64; as cenas em casa e na escola evocam flexibilidade, facilidade de uso e o valor educativo do BBC Micro; já o cenário do Apple Macintosh, mais para a ficção científica, faz alusões ao livro *1984*, prometendo aos usuários a eliminação do trabalho escravo, num mundo humanizado pelo computador. Entretanto, como acontece na maioria das técnicas de comunicação de massa, essas imagens podem ter efeitos não previstos: o elefante é proverbialmente conhecido pelo seu medo aos ratos (ao contrário do Macintosh, que utiliza mouses) e pode, ainda, sugerir um produto antiquado; nada impede que os compradores em potencial do BBC achem o ambiente familiar sufocante e estereotipado, bem como a vinculação com a escola intimidante; e os possíveis clientes da Apple talvez se sintam retratados como autômatos sem mente

to a um usuário neófito, que provavelmente não compreenderá os detalhes técnicos mais sutis. Os jogos e aplicativos financeiros muitas vezes só revelam suas melhores possibilidades depois de comprados e colocados em uso.

Por esse motivo, os fabricantes de hardware e de software, tal como os de cigarros e de sabão em pó, recorrem à criação de uma imagem do produto. É mais eficaz vender uma idéia que um simples bem de consumo, princípio representado pela seguinte máxima publicitária: "O que se vende é o fritar, não a lingüiça".

O design da máquina (sua aparência externa, o vídeo, a distribuição espacial das teclas — de função e do cursor etc.) é muitas vezes um ponto de partida útil para o desenvolvimento da imagem de um hardware. O gabinete pode estar equipado com dispositivos "aceleradores" e logotipos chamativos para atrair os que apreciam jogos. A linha Atari XL — que modificou o estilo, em 1983, como parte do esforço comercial para salvar o produto durante uma fase ruim de vendas — possui design retilíneo, austero, e aberturas para ventilação que enfatizam sua aparência "militar" (bem apropriado para se jogar Batalha de Tanques).

A Commodore, em sua luta pelo mercado americano, incumbiu nada menos que Ferdinand Porsche, o projetista do famoso automóvel, de realçar seus produtos com um desenho em linhas elegantemente arredondadas, com o objetivo de parecer futurista, mas não ameaçador.

Por outro lado, a aparência do Spectrum da Sinclair, com seu gabinete pequeno e teclas de múltiplas funções elaboradamente inscritas, evoca uma máquina que consegue compactar muito num pequeno espaço, sugerindo uma boa compra por baixo preço.

A imagem do produto vai muito além de considerações quanto ao aspecto físico da máquina e precisa apoiar-se numa campanha publicitária coerente que reforce a idéia. A suposta memória "jumbo" do Commodore 64 é sugerida pela presença de um elefante nos comerciais de televisão e anúncios de revistas. Para o BBC, o logotipo da coruja evoca no público a idéia de sabedoria.

Naturalmente o nome é algo da maior importância. Os primeiros micros tiveram de combater uma imagem passada ao público pelos filmes de ficção dos anos 60 e início dos 70, em que o computador era descrito como um desumano Grande Irmão — o ditador do livro *1984*, de George Orwell, que projeta um futuro sombrio no qual a sociedade é controlada por um poder centralizado. Essa a razão de os micros terem recebido nomes familiares, afastando deliberadamente sua vinculação com a alta tecnologia. Daí o PET (mascote) da Commodore, e o Apple (maçã, mas que se presta a vários trocadilhos). Em 1984, o Macintosh (nome de um tipo especial de maçã) foi lançado na Inglaterra e nos Estados Unidos por intermédio de um provocativo anúncio de televisão, ganhador de vários prêmios Clio (o Oscar da publicidade). No filme, uma tela de tevê com a figura do Grande Irmão era destruída por uma mulher — esta representando a nova liberdade oferecida pelo mais recente produto da Apple. E o texto acrescentava: "Graças ao Macintosh, 1984 não será como *1984*".

Mas os compradores precisam ter certeza de que não estão adquirindo um brinquedo. Nesse sentido, o PET, da Commodore, enfrentou muita resistência, quando essa indústria tentou entrar no mercado profissional, no final da década de 70. A primeira linha de ataque foi sugerir

Commodore

que PET era a sigla de Personal Electronic Transactor ("administrador eletrônico pessoal"), mas a tática não foi muito persuasiva, e a empresa acabou por rebatizar o micro como Commodore Business Computer ("computador profissional da Commodore").

A alta tecnologia entrou em cena à medida que o público foi se familiarizando com os computadores no trabalho, no lar e na escola. Segundo parecem achar alguns fabricantes, a eficiência dos nomes dos produtos nessa área é diretamente proporcional à ausência de significado. Vale dizer: quanto mais obscuras as siglas ou os conjuntos de letras, maior a impressão provocada. Daí a preferência por estas e não por palavras dicionarizadas. As letras mais raras, de maior valor no jogo de força, também são muito buscadas aqui. Nada a estranhar, portanto, em nomes como ZX81, MTX500 e MZ-700.

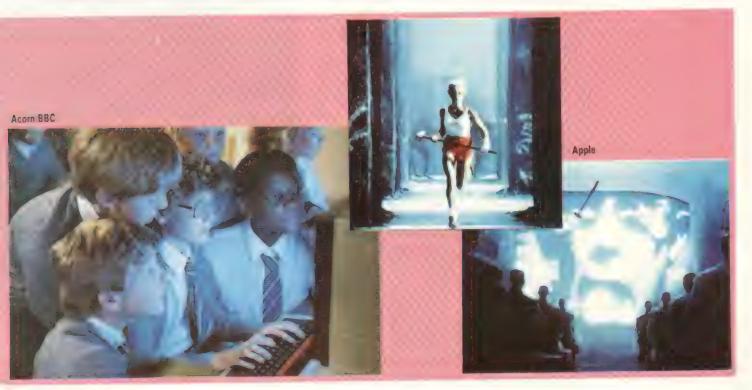
A embalagem, meramente protetora no caso do hardware, assume especial importância para a imagem do software. Em termos gerais, existem duas estratégias para os fabricantes de software: gastar o mínimo na embalagem (a caixa do cassete e um folheto colorido acompanhando) e oferecer um correspondente preço "de ocasião"; ou criar "valor aparente" do produto, acondicionando-o numa caixa com tamanho ampliado (muitas vezes elaborada para que aparente ser um livro) e com o acréscimo de itens adicionais que podem nem fazer parte do jogo. Por exemplo, o jogo Hobbit vem acompanhado por exemplar do livro de mesmo nome, de Tolkien. O jogo de detetive Deadline, que vem numa pasta tipo 007, inclui relatórios policiais, exames de laboratório e até mesmo amostras de materiais encontrados no local do "crime".

A imagem é projetada através de publicidade. Em termos gerais, a relação entre marketing e propaganda é a mesma que entre tática e estratégia. As questões sobre quando usar publicidade e qual seu orçamento são decisões de marketing. O fabricante de micros geralmente lança um novo produto pouco antes da época de Natal e investe boa parte da verba publicitária do ano para apoiar o lançamento.

A promoção da imagem de um produto nem sempre auxilia as vendas. É o que acontece quando a imagem está errada. Uma campanha para cigarros ficou famosa nos meios publicitários: certa marca X recebeu ampla divulgação no cinema e em anúncios de televisão, vinculada a um homem solitário com uma capa de chuva. A chamada era: "Com X, você nunca está sozinho". Mas a mensagem de fato passada foi esta: "Os solitários fumam X", e a marca acabou rejeitada.

Algumas imagens atuais ligadas aos computadores talvez estejam produzindo efeitos negativos da mesma forma. O elefante da Commodore pode servir para lembrar que 64 Kbytes é pouco para o desenvolvimento de programas. Muitos distribuidores do BBC Micro se empenham intensamente em ocultar as fortes vinculações do aparelho com a educação, temendo que tal imagem possa afastar possíveis compradores.

Podem-se considerar o desenvolvimento e a projeção de uma imagem como o lado criativo do marketing; dessa parte se incumbe o pessoal da publicidade. Mas a logística da comercialização tem igual importância — talvez mais, por ser, com freqüência, o elo fraco da cadeia. Uma coisa é mexer com a imaginação do público com relação a um novo produto; colocá-lo em suas casas ou escritórios se revela questão bem mais difícil.

Acorn BBC

Apple

SAÍDA PARA A EXPANSÃO

**O Apple e o IBM PC devem boa parte de seu sucesso à modularidade de suas máquinas, expansíveis conforme as necessidades do usuário.
Isso é possível mediante a adição de placas de circuitos integrados, que examinaremos neste artigo.**

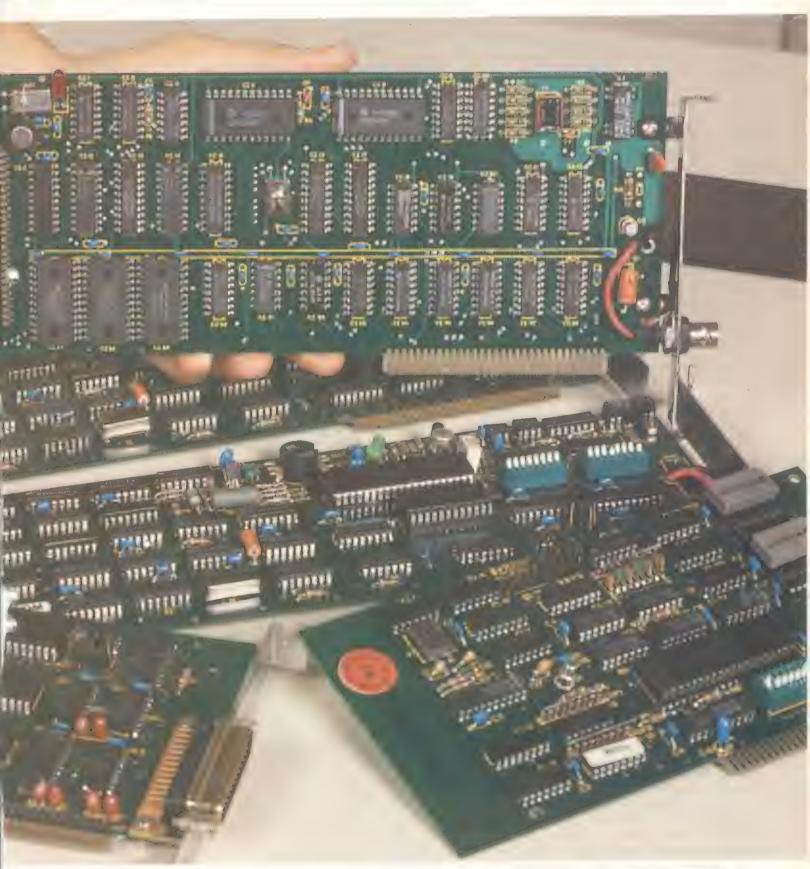
Se você já viu um Apple ou um IBM PC com o gabinete aberto, deve ter observado suas várias placas (também chamadas cartões ou, ainda, interfaces) dispostas verticalmente, encaixadas numa placa principal, maior, fixada na base do equipamento. O que são essas placas e para que servem?

Quando a Apple lançou o primeiro micro, seus projetistas introduziram um conceito inovador: o da máquina "aberta", ou expansível. A configuração básica do equipamento apresentava apenas a placa principal (em inglês mother board, "placa-mãe"), que continha os chips da CPU, controle de vídeo, teclado, som e cassete. Continha também oito saídas ou conectores tipo pente, onde o usuário encaixaria novas placas, à medida que fosse necessitando. Criou-se assim o conceito de hardware modular, inovador em relação ao predecessor do Apple, o TRS-80 — uma máquina "fechada", que não admite acréscimos feitos pelo usuário.

A modularidade garantiu o sucesso do Apple, pois, com um investimento inicial baixo, o usuário adquire o equipamento básico, aumentando passo a passo a capacidade e versatilidade de seu micro por meio de placas que exercem diferentes funções.

Quando a IBM resolveu entrar no mercado de micros, sua equipe de desenvolvimento não poderia deixar de aproveitar esse conceito de modularidade, já utilizado por ela nos computadores de grande porte. Assim, dotaram com cinco conectores para expansão o IBM PC, que, em relação ao Apple, é uma máquina mais profissional, com processador mais potente e maior capacidade de memória. Porém a IBM cometeu um erro de subdimensionamento, pois, das cinco saídas, duas já vinham ocupadas, na configuração básica, pelas indispensáveis placas controladoras de disco e de vídeo. Dando-se conta que as três restantes seriam insuficientes para as necessidades dos usuários, seu lançamento posterior, o IBM XT, veio com oito saídas, das quais cinco ficam disponíveis para o usuário (o IBM XT ocupa mais uma saída com a placa controladora de discos rígidos).

Tanto no Apple como no IBM PC algumas placas são indispensáveis para que o micro exerça suas funções. A mais útil é a expansão da me-

mória, com a qual se executam programas maiores e mais complexos do que o permitido pelos chips de RAM existentes na placa principal. Há expansões de 16 até 128 Kbytes para o Apple; para o IBM PC, essas placas atingem 256 Kbytes.

Quase indispensável também é a placa controladora da impressora, sem a qual torna-se impossível utilizar esse periférico. As impressoras paralelas exigem uma placa especial, ao passo que as seriais podem valer-se da serial, que tam-

Conceito revolucionário
Com as placas, os micros adquiriram extraordinária versatilidade, adaptando-se às necessidades específicas de cada usuário.

Placa de comunicação
A placa NCT 7101, fabricada pela empresa brasileira NCT, transforma um IBM PC num terminal IBM 3278/3279, capaz de comunicação interativa com um computador central.

bém serve para a comunicação com outros micros e acesso a videotextos e a bancos de dados públicos, como, por exemplo, o Cirandão. A placa serial normalmente possui o padrão RS232C, e sua função básica é serializar/desserializar os dados transmitidos ou recebidos.

Ainda no campo das comunicações, há placas especiais para a formação de redes de micros, e, especialmente para o IBM PC, existe uma variada gama de placas que permitem a esse aparelho acessar equipamentos de grande porte. Também o modem, utilizado para comunicação à distância, apresenta-se às vezes sob a forma de uma placa, em vez da conhecida caixinha plástica externa ao micro.

Para aplicações gráficas, também há placas especiais; no caso do IBM PC, permitem aplicações sofisticadas, como CAD/CAM e o uso de até 256 cores (dezesseis básicas em dezesseis tonalidades).

No processamento profissional de textos, bem como na utilização de planilhas financeiras, é necessário que o vídeo do Apple dobre sua apresentação normal, que é de 40 colunas. Para isso utiliza-se a placa de 80 colunas, coisa que o IBM PC dispensa, pois já incorpora essa opção.

Quanto aos sistemas operacionais, o popular CP/M, inerente à arquitetura do IBM PC, também pode ser incorporado a um Apple mediante placa contendo o processador Z80 — também chamada Softcard, pois vem acompanhada de um disco com o software do sistema operacional CP/M. Dotado desse recurso, o Apple roda uma grande quantidade de programas aplicativos, passando seu processador original (o 6502) a funcionar como co-processador auxiliar.

Que dizer dos recursos de síntese de som e voz, tão populares nos jogos e aplicações amadorísticos? No mercado internacional, uma grande variedade de placas sintetizadoras de som transforma um Apple num instrumento musical de recursos nada desprezíveis, pois, além de tocar, memoriza as melodias. Quanto à síntese de voz, recurso que vai se tornando cada vez mais sofisticado, existem placas cujos chips armazenam enormes quantidades de fonemas (os elementos básicos das palavras), possilitando variar o volume e o timbre das frases enunciadas. Contudo, ainda não foi resolvido satisfatoriamente o problema da fala entrecortada e do sotaque.

O IBM PC conta ainda com placas de controle de processos, usadas na automação industrial; conversoras AD/DA (analógico-digital/digital-analógico), muitos tipos de placas de comunicação, e ainda uma que possibilita cópia para fitas de videocassete de arquivos armazenados em discos rígidos, criando assim as indispensáveis back-ups (cópias de segurança).

Quem produz todas essas placas? Os fabricantes dos micros costumam produzir as essenciais — por exemplo, expansões de memória e controladores de vídeo e de discos. As outras, opcionais, provêm de dezenas de fabricantes do mercado internacional, alguns rivalizando em volume de vendas com os próprios fabricantes de equipamentos. No Brasil havia, em 1985, três grandes fabricantes de placas para o Apple, e apenas um — a NCT — servindo aos micros da IBM.

Além dessas duas grandes marcas, a linha Sinclair (no Brasil, TK) também possibilita a adição de algumas expansões opcionais, mas isso é feito por cartuchos, que se encaixam na traseira da máquina. Já os micros do padrão MSX, lançado no Brasil em 1985, refinaram um pouco esse processo, oferecendo também cartuchos conectores, que dão saída para outras expansões; esse sistema, contudo, não possibilita tanta versatilidade quanto a oferecida pelas placas de circuitos integrados.

CONSULTE O ESPECIALISTA

Nos anos 70, a pesquisa sobre inteligência artificial voltou-se para a codificação do conhecimento humano. Surgiram então os sistemas especialistas, dotados da qualificação técnica e profissional dos experts.

Procura-se um especialista
Os sistemas computacionais que assimilam o conhecimento de experts e utilizam esse conhecimento para emitir orientação ou diagnósticos tornam-se cada vez mais populares em muitas áreas — medicina, agricultura, arquitetura. Os sistemas especialistas funcionam no âmbito de uma área restrita e fornecem aos profissionais um serviço de consulta em linha direta para ajudá-los no trabalho, sintetizando a experiência das mais altas autoridades num determinado assunto.

Um especialista — um clínico, um geólogo, um agrônomo etc. — é alguém que dedicou muito tempo ao estudo e aprendizado das técnicas que o habilitam a dominar todos os aspectos de seu trabalho. Mas essa pessoa pode ter problemas subjetivos, que interferem em seu desempenho; e, também, mais cedo ou mais tarde morre, perdendo-se boa parte de seu conhecimento. Na década de 70, alguns pesquisadores da inteligência artificial (IA) decidiram adiar a busca de máquinas inteligentes e empreender a codificação das informações dos especialistas humanos, a fim de preservá-las.

O conceito de sistema especialista representou um dos primeiros exemplos da IA aplicada, voltada para a solução de problemas bem delineados do mundo real. Suas técnicas logo ultrapassaram os limites dos laboratórios de pesquisa onde foram criadas. Hoje existem sistemas que substituem seres humanos especializados em diagnósticos médicos, classificação de doenças agrícolas e muitas outras atividades. Vamos ver como funcionam.

Um sistema especialista baseia-se num extenso corpo de conhecimentos a respeito de uma área específica de problemas. Em geral, esse conhecimento está organizado como um conjunto de regras que permite ao sistema extrair conclusões a partir dos dados ou premissas fornecidos, podendo oferecer, desse modo, tanto orientação como decisões inteligentes. Essa abordagem baseada em conhecimento representa um avanço de conseqüências revolucionárias na ciência da computação. Ela substitui a fórmula tradicional

dados + algoritmo = programa

por uma nova arquitetura centrada em um "banco de conhecimentos" e um "mecanismo de inferência", de modo que

conhecimento + inferência = sistema especialista.

O que vem a ser um sistema especialista? A lista de verificação das características típicas dada a seguir poderá ser útil.

- Um sistema especialista refere-se a um campo relativamente restrito do conhecimento.
- Deve manipular até mesmo dados incertos e regras não confiáveis.
- Deve poder explicar sua seqüência de raciocínio de modo abrangente e compreensível.
- Os mecanismos de inferência são bem distintos dos fatos armazenados: o conhecimento não está embutido nos procedimentos dedutivos.
- É projetado para expandir-se por acréscimo.
- Baseia-se tipicamente em regras.
- Emite orientação como resultado — e não gráficos ou tabelas de números.

A palavra-chave é conhecimento. Sabe-se que o objetivo de um sistema inteligente para solução de problemas consiste em eliminar a busca cega ou aleatória. Para isso, deve explorar a mesma vantagem que o especialista humano tem em relação ao principiante, ou seja, o conhecimento organizado sobre fatos, sobre regras de inferência e estratégias de solução. Existem quatro componentes num sistema especialista totalmente desenvolvido:

1. O banco de conhecimentos;
2. O mecanismo de inferência;
3. O módulo de aquisição de conhecimento; e
4. A interface de explicação.

Esses módulos são fundamentais. Um sistema baseado em conhecimento pode não ter algum deles, mas um sistema especialista efetivo preci-

O sistema esquematizado
Um sistema especialista abrange vários módulos que permitem passar ao usuário o conhecimento de um expert. Em primeiro lugar, o conhecimento deve ser adquirido e incorporado a um banco de conhecimentos. Para fazer previsões, dar orientação ou fornecer um diagnóstico, o sistema deve então ser capaz de extrair inferências do banco de conhecimentos. Finalmente, a interface de explicação permite ao usuário comunicar-se com o sistema para consultá-lo.

sa de todos. Vamos examinar cada um e explicar como interagem.

Conhecimento e inferência

Os componentes fundamentais do sistema especialista são o banco de conhecimentos e o mecanismo de inferência. No primeiro módulo, as informações armazenadas não constituem o conjunto passivo de registros e itens que se encontram num banco de dados convencional. O banco de conhecimentos contém representações simbólicas da experiência e das regras de julgamento dos especialistas, sob uma forma que permite ao mecanismo de inferência realizar deduções a partir delas.

As principais dificuldades ao se desenvolver um banco de conhecimentos são a representação e a aquisição do conhecimento referente a itens de natureza geralmente não matemática. O problema consiste em como codificar conhecimentos de modo que o computador possa usá-los. Em geral, os seguintes elementos devem estar representados: termos técnicos da área (o jargão dos especialistas); relações estruturais (as interconexões dos componentes); e relações causais (relações de causa e efeito entre os componentes).

A tarefa do engenheiro de conhecimento (uma nova profissão) consiste em selecionar meios apropriados de armazenar essas informações simbolicamente. Foram desenvolvidos quatro métodos:

- Regras na forma SE-ENTÃO (IF-THEN). A condição especifica algum padrão, e a conclusão pode ser uma ação ou uma afirmação.
- Redes semânticas. Representam as relações entre os objetos na área (por exemplo, a de que a baleia é um mamífero) por meio da ligação entre módulos.
- Estruturas genéricas. Podem ter valores pré-assumidos e ações codificadas como valores para casos ou áreas específicas.
- Sentenças de Horn. É uma forma de lógica predicativa em que se baseia a linguagem PROLOG muito empregada em IA e com a qual se podem executar inferências.

Uma regra tirada do sistema Mycin para diagnosticar infecções sangüíneas fornece um exemplo típico da estrutura SE-ENTÃO.

SE:
1. A infecção que requer terapia é meningite; e
2. O tipo de infecção é por fungos; e
3. Não forem observados organismos na lâmina de cultura; e
4. O paciente não for um hospedeiro comprometido; e
5. O paciente esteve em uma região em que as coccidiomicoses são endêmicas; e/ou
6. O paciente for indiano, asiático ou de raça negra; e
7. O antígeno criptocócico no exame não for positivo;

ENTÃO
Há evidências sugestivas de que o criptococo não é um dos organismos que poderiam estar causando a infecção.

A estrutura SE-ENTÃO usada pelo Mycin consiste basicamente numa série de instruções identificáveis como verdadeiras ou falsas. As instruções podem ser vinculadas por meio de operadores booleanos como E e OU, para auxiliar a manipulação pelo computador. Para extrair as informações necessárias ao diagnóstico, o Mycin deve dialogar com o usuário. Este deve ter (nesse caso, pelo menos) algum conhecimento médico para compreender e responder às perguntas do sistema especialista.

Mecanismos de inferência

Os mecanismos de inferência são métodos de busca e raciocínio que permitem ao sistema encontrar soluções e, se necessário, fornecem justificativas para suas respostas. Há duas estratégias gerais de raciocínio: encadeamento para a frente e encadeamento para trás. O primeiro envolve raciocínio a partir do observável (sintomas) em direção às conclusões (diagnóstico). Num sistema baseado em regras, isso envolve apenas a comparação entre as condições e os fatos, possivelmente numa ordem predeterminada. O encadeamento para a frente é fácil de ser computadorizado, mostrando-se adequado nos casos em que todos os dados serão, de qualquer forma, coletados. Exemplo disso encontramos nas

situações em que os dados são gerados de modo automático por um instrumento e em que se deve preencher um questionário.

O encadeamento para trás funciona a partir da hipótese em direção ao observável. O sistema escolhe uma hipótese e busca dados que a apóiem ou a refutem. Pode ser programado de modo recorrente e, nos sistemas tipo consulta, leva geralmente a um tipo de diálogo mais natural. A questão de que hipótese escolher numa dada situação não foi até agora resolvida a contento; na prática, a maioria dos sistemas utiliza uma mistura de encadeamento para a frente e para trás.

O módulo de aquisição

Os especialistas têm notável dificuldade em explicar como chegam a suas conclusões, não por desejarem conservar segredos da profissão, mas porque muitos de seus processos de pensamento se dão abaixo do consciente, a nível intuitivo. Assim, a aquisição de conhecimento veio a ser considerada o principal obstáculo ao desenvolvimento de sistemas especialistas. Os experts, porém, tendem a ser bons críticos. Podem examinar um caso-exemplo e dizer que decisões devem ser tomadas e, se necessário, criticar a solução sugerida pelo computador. Esses aspectos favorecem o desenvolvimento de ferramentas de software que permitem a um sistema especialista induzir seus próprios conhecimentos a partir de exemplos pré-classificados. Além de auxiliar o processo de aquisição de informações, essa capacidade permite superar muitas das dificuldades de extrair conhecimentos de especialistas humanos e da laboriosa tarefa de codificá-lo para o computador. Mesmc que o sistema não possa fazer sozinho todo o trabalho — a montagem de um banco de dados de exemplos ou de um conjunto de regras de decisão —, será útil se puder elaborar seu próprio banco de conhecimentos durante um período de "aprendizado" ou durante a utilização, aprendendo a partir dos próprios erros.

A interface de explicação

Uma das boas qualidades do Mycin, do Prospector e de muitos outros sistemas especialistas clássicos está em poderem justificar, quando solicitados, suas conclusões. Evidentemente, quando se passam ao computador responsabilidades envolvendo grandes lucros e perdas ou questões de vida e de morte, é essencial que o sistema explique seu raciocínio. Eis um exemplo de como o Mycin responde a uma pergunta POR QUÊ?, colocada pelo usuário (que, sendo um médico, não aceitará diagnósticos sem explicações). O diálogo começa pela solicitação de um dado ao Mycin.

A meningite é uma infecção adquirida em hospital?
POR QUÊ?
[2.0] Isso auxiliará a determinar os organismos (além dos observados em culturas ou lâminas) que poderiam estar ocasionando a infecção. Já foi determinado que:
[2.1] a infecção que exige terapia é meningite; e
[2.2] não foram observados organismos na lâmina dessa cultura em exame; e
[2.3] o tipo de infecção é bacterial; e
[2.4] a neurocirurgia não é um dos itens relevantes no histórico desse paciente; e
[2.5] a idade do paciente está acima de dez dias.
Portanto, se:
[2.6] a infecção foi adquirida enquanto o paciente estava hospitalizado, então há evidências de que os organismos (além dos observados nas culturas e lâminas) que poderiam estar causando o quadro infeccioso são *E. coli* (0,75), estafilococos-coag-pos (0,3), pseudomonas-aeruginosa (0,3), Klebsiella-pneumoniae (0,5)
— baseado em [REGRA 545].

Esse recurso de explicação é, na verdade, um acompanhamento parcial do processo de raciocínio do programa especialista, expresso no jargão médico e em linguagem comum. Tais explicações podem ser fornecidas com relativa facilidade e a baixo custo, em sistemas baseados em regras (pela apresentação das utilizadas e suas predecessoras), o que representa um ponto a favor da codificação do conhecimento baseada em regras.

Observe que as conclusões vêm acompanhadas por pesos numéricos. A rigor, esses números não expressam probabilidades. São pesos que permitem ao sistema lidar com a incerteza de modo coerente e apresentar uma lista ordenada de possíveis diagnósticos no final da análise.

Orientação pelo videotexto

O sistema especialista destinado aos agricultores ingleses é acessado pela rede de videotexto...

A introdução do programa permite ao usuário escolher a orientação de planejamento para a meia-estação ou para o outono...

O sistema dirige ao agricultor uma série de perguntas sobre seu plantio, nas quais baseará sua orientação...

Tendo interpretado as respostas do agricultor, o sistema fornece a orientação desejada sob a forma de valores de avaliação para vários tratamentos químicos possíveis.

O cérebro dos cereais
O Wheat Counsellor (Conselheiro do Trigo) é um sistema especialista posto gratuitamente à disposição dos lavradores através da Prestel, o sistema britânico de videotexto. Foi projetado para fornecer orientação sobre métodos de combate a pragas do trigo, adequada aplicação de defensivos e estimativa de perdas. O sistema simula o tipo de conversa que poderia ocorrer entre fazendeiros — mas seu conhecimento foi acumulado a partir de estudos científicos sobre a disseminação de doenças agrícolas. O sistema "extrai" do agricultor as informações necessárias por meio de perguntas simples. A orientação é transmitida sob a forma de uma lista de tratamentos possíveis, juntamente com uma avaliação de cada um.

ATARI XL

A Atari entrou bem cedo no mercado dos micros, com os modelos 400 e 800. Seus dois novos lançamentos são mais baratos e incluem refinamentos muito bem recebidos.

A Atari respondeu à competição crescente no mercado de micros lançando o 600XL e o 800XL, que podem usar todo o software das versões anteriores, 400 e 800.

Os novos teclados, bem desenhados e confortáveis, têm 62 teclas, 29 das quais acionam funções gráficas. As outras abrigam o conjunto completo dos caracteres ASCII. Tal como no 800, quatro teclas com flechas movimentam o cursor: basta apertar [CONTROL] e a tecla desejada.

Todos os micros da Atari podem ser ligados a um aparelho de tevê. A resolução gráfica é boa, bem como o contraste, no modo texto, entre as palavras e o fundo. Há várias cores disponíveis. Os XLs possuem onze modos gráficos e até 256 cores (dezesseis cores com dezesseis tonalidades cada). Devido à quantidade de memória necessária para formar uma tela, o número de cores depende da resolução: esta, quanto maior for, menos cores poderão ser exibidas.

Quanto ao som, os XLs dispõem de um chip especial que possibilita até quatro vozes independentes, cada uma podendo cobrir 3 1/2 oitavas. São controladas em BASIC pelo comando SOUND, ou por POKES. Podem-se manipular a oscilação, a altura, a distorção e o volume de cada som.

O 400 e o 800 não vêm com linguagem residente. Os XLs, no entanto, possuem um BASIC, mas ele é bastante deficiente. Quem quiser um melhor terá de recorrer aos cartuchos.

Com uma placa de expansão, os 16 Kbytes de memória disponíveis do 600XL atingem até 64 Kbytes. Talvez o periférico mais útil já projetado pela Atari para os XLs seja a caixa de expansões. Comporta oito aberturas de expansão, duas saídas seriais padrão RS232 e um bus paralelo. Há também um módulo CP/M, com microprocessador Z80.

Um dos pontos fortes desses micros está em sua grande biblioteca de aplicativos em cartuchos, cassetes e disquetes, que vão de jogos a pacotes de aplicações comerciais.

A impressora Atari 1027
Com um cabeçote parecido com as esferas usadas em máquinas de escrever, proporciona uma impressão tão boa quanto a destas, apenas um pouco lenta. Há mais duas impressoras Atari: uma usa canetas hidrográficas para desenhar letras e linhas em quatro cores; a outra, matricial, é rápida, mas imprime numa qualidade inferior. Só essas duas podem ser usadas diretamente com os XLs, devido à ausência de interface padrão para impressoras.

Os gêmeos da Atari
Os microcomputadores 600XL e 800XL são muito parecidos, mas o primeiro tem 16 Kbytes de memória e o segundo, 64 Kbytes. O teclado é de alta qualidade, e os recursos gráficos coloridos mostram-se muito bons. Por serem versões atualizadas dos computadores Atari originais, dispõem também de ampla gama de aplicativos.

Entrada para cartucho
Os modelos XLs têm um único encaixe para cartuchos de aplicativos.

Chips gráficos
Dois chips feitos sob medida, conhecidos como ANTIC e GTIA, proporcionam a excelente capacidade gráfica dos Atari.

RAM
Os 16 Kbytes de RAM compreendem vários chips.

ATARI 600XL/800XL

DIMENSÕES

600XL: 380 x 170 x 40 mm.
800XL: 380 x 220 x 40 mm.

CPU

6502. 2MHz.

CLOCK

2 MHz

MEMÓRIA

16 Kbytes de RAM, expansível até 64; 24 Kbytes de ROM.

VÍDEO

Modo texto:
até 24 linhas x 40 colunas;
Modo gráfico:
até 320 x 192 pixels, com dezesseis cores, em dezesseis tonalidades cada

TECLADO

Estilo máquina de escrever, com 62 teclas, inclusive de movimentação do cursor e para funções exclusivas, tais como [Select], [Start] e [Help], para o controle de programas.

INTERFACES

Joysticks (duas), saída para periféricos, caixa de expansões, entrada para cartuchos.

LINGUAGENS

BASIC, FORTH, LOGO, PILOT e ASSEMBLY do 6502.

DOCUMENTAÇÃO

Os manuais nunca foram o ponto forte da Atari, pois ela tende a encarar seus micros, antes de mais nada, como equipamentos para jogos, restringindo-se a detalhes técnicos. Encontra-se à venda, no entanto, enorme variedade de manuais e revistas independentes.

ROM
O interpretador BASIC residente é mantido nestes dois chips.

Saída para periféricos
Saída de treze pinos para conectar periféricos, inclusive unidade de discos, impressoras e o gravador cassete exclusivo.

Saídas para joysticks

Chip de som
Um chip de fabricação exclusiva — o POKEY — encarrega-se da geração de sons.

Chip de E/S
Um 6520 manipula as conexões de entrada e saída.

CPU
Os Ataris baseiam-se no microprocessador 6502.

Gravador cassete Atari
Os Ataris só funcionam com seu próprio gravador cassete. Embora isso encareça o sistema, apresenta algumas vantagens. Primeiro porque um gravador fabricado pela própria Atari é mais confiável. Segundo porque ele usa duas pistas. Numa, gravam-se programas da maneira habitual; na outra, gravam-se sons. Isso permite, por exemplo, que programas de ensino de línguas apresentem o trecho da fala no momento certo.

OPERÁRIO PADRÃO

Os robôs mais caros do mercado enquadram-se em duas categorias: os que auxiliam na educação, demonstrando os princípios da robótica, e os que representam a perfeição máxima do moderno projeto de robôs.

A ficção científica quase sempre viu nos seres mecânicos a materialização do mal, bestas tecnológicas em busca do poder absoluto pela destruição total. No cinema, é recente o surgimento do indestrutível andróide de *O exterminador do futuro,* criado numa civilização do futuro para retornar ao nosso tempo presente e cumprir sanguinária missão. Essa visão talvez tenha em sua origem uma antiga discussão sobre o uso que o homem faz da máquina ou, antes, o medo de que a criatura se volte contra o criador.

Mas os robôs do mundo real ainda estão longe de sentir e pensar como o homem. No acelerado processo de automação industrial, deixaram até mesmo de receber aqueles traços antropomórficos que os caracterizavam de forma quase lendária. Na verdade, somente hoje, com o desenvolvimento do micro processador, é que se tornou realidade a manipulação programável dos robôs, eliminando assim a necessidade do controle humano a distância.

De uma coisa não há dúvida: o robô é um eficiente e incansável operário, apto a desempenhar tarefas repetitivas, enfadonhas, perigosas ou insalubres para o ser humano. Isso trouxe de volta o espectro do desemprego, que, supunha-se, seria inevitável com a automação industrial. O fantasma não chegou a ser exorcizado, nem mesmo com a experiência positiva do Japão, que, com o extensivo emprego de robôs, produziu, na década de 80, 11 milhões de automóveis com os mesmos 450.000 empregados da década anterior. No Brasil, embora se temessem os prováveis nocivos efeitos sociais, a SEI (Secretaria Especial de Informática) divulgou, em junho de 1985, a lista das dezesseis empresas nacionais autorizadas a fabricar robôs e a desenvolver sistemas de robótica.

Apesar do estágio incipiente das pesquisas, alguns órgãos estão desenvolvendo projetos nessa área. A Universidade Federal do Espírito Santo, por exemplo, desenvolveu um robô com cinco

Às ordens
Por ser móvel e possuir pinças, o Hero é capaz de levar café da manhã ao usuário (desde que não haja degraus entre a cozinha e o quarto, claro). Na ilustração, Hero e Mentor saboreiam um delicioso chá.

O longo braço do aprendizado

O projeto de chips e a robótica constituem hoje a vanguarda da pesquisa microeletrônica. Nem todos os amadores dispõem de recursos para experimentar novos microchips em suas oficinas domésticas, mas já se fabricam robôs bastante acessíveis. Objetiva-se, com esses braços mecânicos, adquirir um conhecimento mais amplo sobre o funcionamento dos robôs. E a melhor maneira de se conseguir isso é pela observação de sistemas em funcionamento e pela tentativa em aperfeiçoá-los. O Micro Grasp, da Powertran Cybernetics, e o Armdroid, da Colne Robotics, são braços programáveis por microcomputadores e têm cinco graus de liberdade. Podem ser adquiridos prontos ou na forma de kits para montar.

graus de liberdade, 1 kg de capacidade de carga e movimentos com precisão de 0,2 mm. Dois micros de 8 bits fazem o controle: um comunica a tarefa aos atuadores de saída e coordena o controle de potência, movimentos, fluxos e verificação da tarefa. O outro liga o braço robótico aos periféricos. Também a Escola Politécnica da Universidade de São Paulo desenvolveu um robô com sensores ópticos e células fotoelétricas.

Dos projetos aprovados pela SEI, a grande maioria utiliza tecnologia própria; apenas uns poucos dependem de patentes estrangeiras. Segundo a classificação japonesa, cerca de metade das empresas trabalham com robôs manipuladores manuais, de seqüência fixa e variável; outras, com robôs repetidores e de controle numérico. Ao que parece, o desenvolvimento de robôs inteligentes, capazes de retroalimentação (feedback), não faz parte das prioridades definidas.

O império dos robôs

Dentre os robôs inteligentes, existem, no mercado internacional, aqueles que não se classificam como industriais, pois são projetados para uso educacional e doméstico. O primeiro grupo compreende os fabricados com objetivo de alcançar o máximo de versatilidade e englobam várias das características dos braços mecânicos industriais. Distingue-os a finalidade: ensinar os princípios da robótica. As maiores diferenças entre esses braços e os industriais residem no fato de aqueles terem uma dimensão menor e inferior capacidade de manipular objetos grandes.

Em muitos casos, a finalidade educacional coincide com a industrial, pois, quando a aplicação fabril exige um braço relativamente pequeno e leve, muitos dos educativos acabam servindo às finalidades industriais. Um robô industrial, por exemplo, talvez seja indispensável à manipulação de enormes lingotes pesando centenas de quilogramas tanto quanto para a montagem de componentes em delicadas placas de circuito impresso — tarefa que dispensa um braço grande e poderoso. Assim, os braços robóticos que se enquadram nessa categoria efetuam aplicações importantes, mas sem deixarem de ser, ao mesmo tempo, educacionais. Essa segunda categoria compreende robôs cujo projeto incorpora as mais recentes conquistas da robótica contemporânea. Muitos são dotados dos dispositivos sensoriais examinados nos artigos anteriores desta série.

Braços educadores

Um exemplo de robô desse tipo é o Mentor, da Cybernetic Applications. Vendido em kit, permite seis graus de liberdade (cintura, ombro, cotovelo e três eixos de rotação no pulso) e se movimenta por motores elétricos. Seu controle pode ser feito pelo BBC Micro, pelo Commodore Vic-20 ou pelo Sinclair Spectrum.

Também encontráveis na forma de kit temos o Neptune 1 e o Neptune 2. Esses braços mecânicos levantam até 2,5 kg e são articulados por um sistema hidráulico, embora usem água e não o fluido hidráulico em geral empregado. Tam-

bém podem ser controlados pelo BBC Micro, pelo Vic-20 e pelo Spectrum. As duas velocidades de operação do Neptune 2 constituem uma vantagem, pois os braços se movem com rapidez, quando são necessários movimentos amplos, ou mais devagar, no caso de trabalhos de precisão.

A Powertran Cybernetics fabrica o Genesis P101, com seis graus de liberdade e vendido em kit. Movido hidraulicamente, esse modelo vem equipado com uma caixa de controle para a programação do robô, além de uma interface RS232, que permite sua conexão à maioria dos computadores. Este braço também pode ser adquirido já testado e pré-montado, mas a um preço maior.

Um outro braço mecânico, também pré-montado, é o Cyber 310, da Cyber Robotics. Há versões disponíveis para o BBC Micro, o Jupiter Ace, o Apple II, e os Commodores PET 3000/4000 e 8000. O braço, movido por motores graduais, tem capacidade de elevação de apenas 250 g. Isso o caracteriza como um peso-leve, mas oferece, ainda assim, uma impressionante faixa de aplicações. Além de cinco graus de liberdade, sua velocidade pode ser acelerada e desacelerada pelo usuário, simulando o movimento do braço humano. Outro recurso disponível é a alteração constante da velocidade, de acordo com a natureza da tarefa. Podem-se mover todas as articulações simultaneamente e especificar a posição relativa (por exemplo, ordena-se ao braço que se mova para a frente xis unidades a partir da posição atual) ou absoluta do braço (especifica-se um movimento até certo ponto em relação à posição de "repouso"). É programável em BASIC e também numa versão do FORTH conhecida como Robo FORTH, desenvolvida pela Cyber Robotics.

Há ainda o HRA933 e o HRA934, da Feedback Instruments. O primeiro é mais acessível ao amador, e ambos vêm pré-montados. Dispõem de um sistema de operação que permite

NOME	NEPTUNE 1	NEPTUNE 2	GENESIS P101	HERO
TIPO	Braço mecânico	Braço mecânico	Braço mecânico	Robô de solo
PROPÓSITO BÁSICO	Educacional	Educacional	Educacional	Educacional e experimental
SENSORES	Um potenciômetro registra a posição; a pinça detecta grau de fechamento	Um potenciômetro registra a posição; a pinça detecta grau de fechamento	Realimentação posicional	Ultra-sônicos, permitindo detectar movimento. Detectam 256 níveis de luz e som. Sensores táteis na pinça
GRAUS DE LIBERDADE	6: cintura, ombro, cotovelo, elevação do pulso, giro do pulso, pinça	7: cintura, ombro, cotovelo, elevação do pulso, giro do pulso, pinça	6: cintura, ombro, cotovelo, elevação do pulso, giro do pulso, pinça	4: ombro, cotovelo, pulso, pinça
ALIMENTAÇÃO	Hidráulica: bomba de água acionada pelo cabo de alimentação	Hidráulica: bomba de água acionada pelo cabo de alimentação	Hidráulica: bomba de água acionada pelo cabo de alimentação	Baterias recarregáveis
CONECTA-SE AO	BBC Micro, Spectrum, Vic-20	BBC Micro, Spectrum, Vic-20	Programável por uma caixa controladora; interface padrão RS232; BBC Micro, Spectrum, Vic-20	Usando um conector em cruz, pode ser acoplado a qualquer micro com saída serial
FABRICANTE	Cybernetic Applications Ltd., Hampshire, Inglaterra	Cybernetic Applications Ltd.	Powertran Cybernetics, Hampshire, Inglaterra	Zenith Data Systems, Gloucester, Inglaterra

cinco graus de liberdade, possibilitando erguer 1,35 kg com 3 mm de precisão. Além dos sensores de posição para as articulações dos braços, possuem também sensores táteis na extremidade dos "dedos". Eles indicam o momento em que um objeto é agarrado, possibilitando que os braços controlem a força aplicada. Uma interface RS232 faz a conexão. Para as instruções de controle, usa-se o Apple II, o Tandy TRS-80 e o Commodore PET.

Perfeição absoluta

O robô Hero-1, da Zenith Data Systems, é vendido em kit a um preço relativamente alto, compensado, entretanto, pelos grandes recursos que oferece. Móvel, seu braço utiliza um sistema de coordenadas esféricas, permitindo-lhe expansão e contração telescópicas. O Hero-1 vem equipado com um conjunto de sensores para detecção de movimentos, sons e luzes, incluindo um sensor ultra-sônico de distâncias, que evita colisões, e um sintetizador de voz, que lhe dá um vocabulário potencialmente ilimitado. Como sua montagem não é simples, muitas vezes é preferível adquiri-lo pronto, mesmo que a um preço mais elevado.

Em alguns aspectos, todos esses robôs são pouco mais que um dispendioso entretenimento, em termos do que podem oferecer ao usuário. Na verdade, servem principalmente como elemento promocional, distribuindo folhetos em estandes de exposições ou demonstrando produtos. Constituem, no entanto, o resultado da mais avançada pesquisa no campo da robótica. Todos dispõem de sensores inteligentes, movem-se com inteligência e têm braços inteligentes. Embora nenhum deles conte com sistema óptico, todos podem "falar" e reagir a determinados sons.

A alucinante evolução da robótica nos dá a certeza de que, ao contrário da rebeldia e maldade lendárias, os robôs serão leais operários industriais.

MENTOR	CYBER 310	HRA933	HRA934
Braço mecânico	Braço mecânico	Braço mecânico	Braço mecânico
Educacional	Educacional	Educacional	Educacional
Um potenciômetro registra a posição; a pinça detecta o grau de fechamento	Nenhum	Capacidade de determinar sua posição; a pinça detecta o grau de fechamento	Capacidade de determinar sua posição; a pinça detecta o grau de fechamento
6: cintura, ombro, cotovelo, elevação do pulso, giro do pulso, pinça	6: rotação da base, do ombro (até 180°), cotovelo, elevação do pulso, torção do pulso, pinça	5: rotação da base, cotovelo, giro do pulso, torção do pulso, pinça	5: rotação da base, cotovelo, giro do pulso, torção do pulso, pinça
Eletrica	Elétrica	Eletrica	Eletrica
BBC Micro, Spectrum, Vic-20	Apple II, BBC Micro, Jupiter Ace, série Commodore PET e Hector HRX	Apple II, TRS-80, série Commodore PET, AIM 65, BBC Micro, MAT 385	Apple II, TRS-80, série Commodore PET, AIM 65, BBC Micro, MAT 385
Cybernetic Applications Ltd.	Cyber Robotics, Cambridge, Inglaterra	Feedback Systems Ltd., East Sussex, Inglaterra	Feedback Systems Ltd.

O SEGREDO DAS OPERAÇÕES

Vimos como a CPU manipula a memória por meio dos registradores e dos diversos modos de endereçamento. Podemos agora entender como se realizam, no Z80 e no 6502, as operações de adição e subtração.

As diferenças entre os processadores Z80 e 6501, em relação à maneira de executar as operações aritméticas simples, demonstram bem as peculiaridades de cada CPU. O Z80, com seus muitos registradores e complexo conjunto de instruções, é um processador potente e sofisticado. Já no 6502, a simplicidade da arquitetura e do conjunto de instruções configura um projeto mais modesto — robusto e prático, mas sem atingir a categoria do Z80. Essa impressão, contudo, mostra-se exata só até certo ponto, pois a variedade de modos de endereçamento do 6502, aliada ao uso da página zero como um registrador de índice adicional, fornece uma versatilidade que lhe permitirá continuar dominando o mercado dos micros de 8 bits ainda por algum tempo.

O Z80, por outro lado, leva vantagem na flexibilidade de seus registradores, que podem ser tratados simultaneamente como possuindo 1 byte ou 2, possibilitando assim grande capacidade de endereçamento. O 6502, por outro lado, não possui registradores de 2 bytes, mas seus diversos modos de endereçamento permitem-lhe tratar a página zero como uma grande matriz de registradores de 1 e 2 bytes.

Aritmética básica

Já vimos que os registradores da CPU permitem acessar a memória de diversas maneiras; contudo, a manipulação da memória não se limita a carregar, armazenar e comparar dados. A execução das quatro operações aritméticas é essencial a qualquer sistema computacional; porém tanto o Z80 como o 6502 efetuam apenas a adição e a subtração. A multiplicação e a divisão necessitam ser programadas, assim como a adição e a subtração de números maiores que $FF. Essa é uma limitação de ambas as CPUs, se bem que, para o programador, a elaboração de algoritmos para essas finalidades seja um valioso exercício. Vale lembrar também que os processadores de 16 bits, sucessores do Z80 e do 6502, executam essas duas operações graças à maior potência e velocidade de que dispõem.

Para executar operações aritméticas de um só byte já utilizamos as instruções ADC ("adição com transporte") e INC ("incrementar"), disponíveis em ambas as CPUs. Eis, nas duas versões, uma adição do conteúdo de duas posições de memória, ambas com 2 bytes:

6502	Z80
ADDR1 DW $7E60	ADDR1 DW $7E60
ADDR2 DW $4A51	ADDR2 DW $4A51
SUM DS $03	SUM DS $03
BEGIN CLC	BEGIN LD A,$00
LDA ADDR1	AND A
ADC ADDR2	LD HL,(ADDR1)
STA SUM	LD DE,(ADDR2)
LDA ADDR1+1	ADD HL,DE
ADC ADDR2+1	LD (SUM),HL
STA SUM+1	ADC A,$00
LDA $00	LD (SUM+2),A
ADC $00	RET
STA SUM+2	
RTS	

O método de um único byte empregado no 6502 é utilizável também no Z80, mas a técnica por este empregada, utilizando um par de registradores, não tem equivalente no 6502. Observe as estratégias que manipulam as diversas possibilidades de transporte, desde as instruções CLC (6502) e AND A (Z80), que limpam o flag de transporte antes da adição, até a modificação do terceiro byte de SUM. Como em qualquer cálculo aritmético, é vital supor o resultado máximo.

O tratamento da subtração assemelha-se ao da adição, uma vez que ambos os processadores possuem a instrução SBC ("subtração com transporte"), embora o Z80 possibilite também a subtração com 2 bytes. A subtração, contudo, pode gerar um resultado negativo e, portanto, devemos estudar agora a representação binária dos sinais algébricos.

Tudo o que precisamos saber, no momento, sobre números negativos está implícito nesta afirmação:

Se $A+B=0$, então $A=-B$

Isso implica que, sendo A um número positivo, sua negação ou complemento é o número que, somado a A, dá como resultado 0. Por exemplo, se A for o número de 1 byte $04, seu complemento de 1 byte será $FC:

$04 + $FC = $100

Lembrando que $100 = $00 (se tivermos um registrador de único byte para armazenar o resultado), essa representação complementar significa que a subtração pode ser considerada como a soma de números negativos, ou seja:

$A-B$ é o mesmo que $A+(-B)$

Assim, $08 − $05 equivale a $08 + (− $05) e

Transferências de registradores de único byte

A

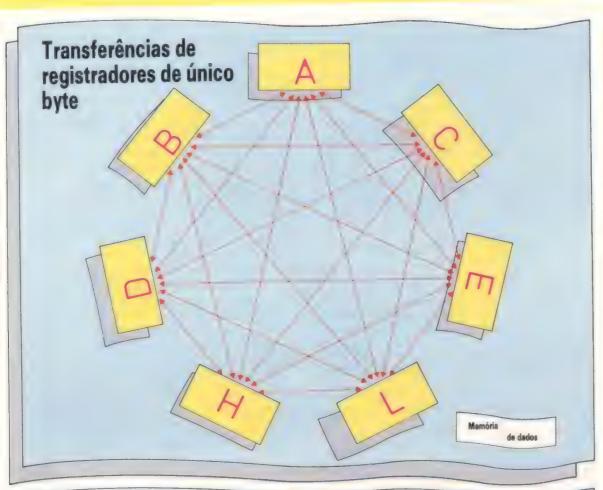
B

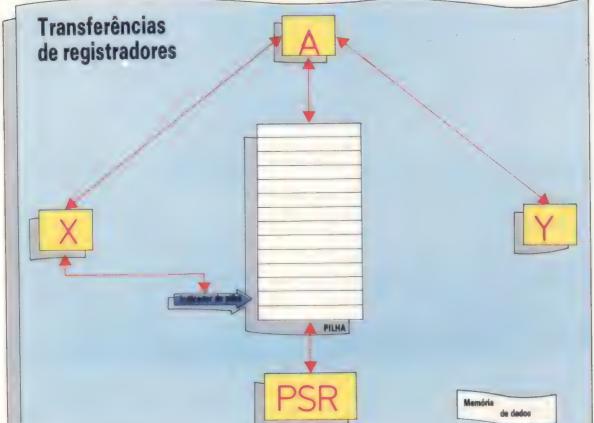
C

D

E

H

L

Memória de dados

Dupla identidade
Os registradores de dados do Z80, de único byte, comunicam-se com qualquer outro registrador de único byte. Cada um deles pode contatar com a memória em modo direto, imediato, indireto, absoluto e indexado. Quando tratados como BC, DE, HL — os pares de registradores de 2 bytes — carregam e transferem dados de 16 bits de memória e da pilha. Funcionam também como acumuladores de 16 bits para a adição e a subtração; essa combinação de recursos e flexibilidade é a chave do grande sucesso do Z80.

Claro e simples
No 6502, a comunicação interna é linear, restrita à transferência de dados de 8 bits. Apenas o acumulador (A) comunica-se diretamente com X e Y; só X comunica-se com o indicador de pilha, e apenas o PSR (Processor Status Regis, "registrador de estado do processador") e o acumulador acessam a pilha. São possíveis transferências na memória nos modos absoluto, direto, indireto, indexado, imediato e página zero. A engenhosa utilização do modo de página zero no 6502 compensa a escassez de registradores, pois a página zero converte-se em 128 registradores de 2 bytes.

(–$05) = $FB (uma vez que $FB + $05 = $100). Portanto, podemos expressar nossa subtração original como $08 + $FB. O resultado dessa soma é $103, ou seja, $03 representado em um só byte.

Esse tipo de representação é conhecido como complemento de dois: forma-se o complemento de um número de um só byte subtraindo-o de $100. Há também outra representação, chamada complemento de um, que se relaciona com a primeira de forma interessante. Veja este exemplo:

```
$05 = 00000101     binário
$FA = 11111010     complemento de um
            +1
-----------------
$FB = 11111011     complemento de dois

$05 + $FA = $FF
$05 + $FB = $00
```

Verificamos assim que, para formar o complemento de um número de único byte, basta complementar — ou seja, negar — cada bit binário do número. Ao somar 1 ao resultado, produzimos o complemento de 2 do número em questão. A soma de um número a seu complemento de 1 sempre totaliza $FF, e a soma de um número a seu complemento de 2 sempre resulta $00 (na verdade, $100). Convencionou-se, portanto, na aritmética dos inteiros relativos, considerar positivos os números de $00 a $7F (de 0 a 127), e negativos de $80 a $FF (–128 a –1). Ao compararmos as representações binárias desses números, notamos que todos os negativos têm o bit 7 em 1 e os positivos sempre 0. Por isso, se conhece o sétimo bit como bit de sinal de um número relativo. Após cada operação aritmética ou lógica, é o sétimo bit do resultado que vai ativar ou desativar o flag de transporte do registrador de estado do processador.

É indispensável compreender todos esses conceitos para trabalhar com números relativos. Uma vez assimiladas suas implicações, no entanto, utilizam-se regras práticas para manipulá-los. No próximo artigo estudaremos essas regras, bem como os algoritmos para multiplicação e divisão.

Respostas dos exercícios anteriores

A) O seguinte programa inverte a ordem do string de caracteres armazenados em LABL1:

6502

```
;
ORIGIN   ORG   $7000
LAST1    EQU   $0D
LABL1    DB    'EIS UMA MENSAGEM'
TERMN8   DB    LAST1
;
BEGIN    LDX   #$FF
         LDA   #LAST1
         PHA
LOOP0    INX
         LDA   LABL1,X
         PHA
         CMP   #LAST1
ENDLP0   BNE   LOOP0
CLRSTK   PLA
;
BEGIN1   LDX   #$FF
LOOP1    INX
         PLA
         STA   LABL1,X
         CMP   #LAST1
ENDLP1   BNE   LOOP1
         RTS
```

Z80

```
         ORG   $C000
LAST1    EQU   $0D
LABL1    DB    'EIS UMA MENSAGEM'
TERMN8   DB    LAST1
;
BEGIN    LD    IX,LABL1-1
         LD    A,LAST1
         PUSH  AF
LOOP0    INC   IX
         LD    A,(IX+0)
         PUSH  AF
         CP    LAST1
ENDLP0   JR    NZ,LOOP0
CLRSTK   POP   AF
;
BEGIN1   LD    IX,LABL1-1
LOOP1    INC   IX
         POP   AF
         LD    (IX=0),A
         CP    LAST1
ENDLP1   JR    NZ,LOOP1
         RET
```

Na versão do 6502, as instruções entre LOOP0 e ENDLOOP0 usam o endereçamento indexado em X num loop que carrega os caracteres, um a um, de LABL1, e os coloca na pilha. Transfere-se primeiro o valor ASCII do caractere indicador de término, para marcar o limite inferior da pilha. O último caractere nela introduzido também é o indicador de término, dessa vez determinado por sua posição como último caractere do string. Assim se conclui o loop e, então, apaga-se em CLRSTK o caractere de término no alto da pilha.

A versão do Z80 usa o modo de endereçamento indireto, com IX para carregar o acumulador a partir da LABL1, e introduz na pilha não só o acumulador como também o registrador de flag. Isso significa que os caracteres do string em LABL1 são intercalados na pilha com valores sucessivos do registrador de estado do processador.

O código entre BEGIN1 e ENDLP1, em ambas as versões, reflete o loop anterior e usa as mesmas técnicas, mas dessa vez extraindo da pilha o string de caracteres na ordem inversa e armazenando-o em LABL1. O loop termina quando encontra o caractere de término no fim da pilha.

Note a importância de compensar as introduções com extrações da pilha. A parte mais difícil do problema consiste em lidar com as condições extremas — como iniciar e terminar os loops, e ajustar os valores e condições que dele dependem.

A instrução de Z80 em BEGIN e BEGIN1 (LD IX, LABL1-1) ilustra a utilidade de um programa compilador. Aqui ele interpreta a expressão (LABL-1) como "o endereço do byte imediatamente anterior àquele cujo endereço é LABL1"; introduz, então, tal endereço no código. A maioria dos compiladores possui algum tipo de avaliação de expressões, em geral permitindo a modificação de um ou dois operandos por um operador aritmético — normalmente "+" ou "−".

B) Este programa inverte a ordem dos caracteres dentro de cada palavra do string em LABL1, mas mantém a ordem das palavras:

6502

```
;
ORIGIN  ORG   $7000
LAST1   EQU   $0D
SPACE   EQU   $20
LABL1   DB    'EIS UMA'
TERMN8  DB    LAST1
;
BEGIN   LDX   #$FF
LOOP0   JSR   RVSWRD
        CMP   #LAST1
ENDLP0  BNE   LOOP0
        RTS
;
;**INVERTER UMA PALAVRA**
LASTCH  DB    $00
LASTX   DB    $00
RVSWRD  TXA
        TAY
        INY
RVSLP0  INX
        LDA   LABL1,X
        PHA
        CMP   #SPACE
        BEQ   CLRSTK
        CMP   #LAST1
ENDRV0  BNE   RVSLP0
CLRSTK  PLA
        STA   LASTCH
        STX   LASTX
RVSLP1  PLA
        STA   LABL1,Y
        INY
        CPY   LASTX
ENDLP1  BNE   RVSLP1
        LDA   LASTCH
        RTS
```

Aqui há vários pontos de interesse, como o uso das instruções JSR e CALL. A sub-rotina RVSWRD tem estrutura semelhante à do programa dado no exercício A, mas inverte somente os caracteres de uma palavra, e não todo o string. Tanto na versão do 6502 como na do Z80, usa-se o registrador indicador (X e IX, respectivamente) para passar à sub-rotina o endereço inicial da palavra. Utiliza-se também o acumulador para devolver ao programa principal o valor do caractere que termina a palavra (um espaço em branco ou um caractere indicador de término do string). Essa técnica de passagem de valores é muito comum na linguagem ASSEMBLY, mas requer cuidado — em especial se o programador estiver habituado a colocar na pilha todos os registradores da CPU ao iniciar cada sub-rotina.

Outra característica significativa é o uso do registrador Y na versão do 6502. Primeiro ele armazena o endereço inicial da palavra, enquanto X atua como indicador no loop da pilha. Em seguida, Y funciona como indicador no loop de "desempilhamento", enquanto X guarda o endereço final da palavra. Utilizamos aqui a palavra "endereço" de maneira imprópria, pois X e Y são registradores de único byte, portanto incapazes de conter um endereço completo. O que eles armazenam é um deslocamento para o endereço LABL1. Por outro lado, os registradores indicadores IX e IY podem conter um endereço completo de 2 bytes.

Na versão do Z80, não se utilizaram IX e IY, mas sim os pares de registradores HL e DE. Tal como os registradores X e Y do 6502, eles armazenam os endereços de início

Z80

```
        ORG   $C000
LAST1   EQU   $0D
SPACE   EQU   $20
LABL1   DB    'EIS UMA MENSAGEM'
TERMN8  DB    LAST1
;
BEGIN   LD    DE,LABL1-1
LOOP0   CALL  RVSWRD
        CP    LAST1
ENDLP0  JR    NZ,LOOP0
        RET
;
;**INVERTER UMA PALAVRA**
LASTCH  DB    $00
RVSWRD  PUSH  DE
        POP   HL
        INC   HL
RVSLP0  INC   DE
        LD    A,(DE)
        PUSH  AF
        CP    SPACE
        JR    Z,CLRSTK
        CP    LAST1
ENDRV0  JR    NZ,RVSLP0
CLRSTK  POP   AF
        LD    (LASTCH),A
;
RVSLP1  POP   AF
        LD    (HL),A
        INC   HL
        LD    A,L
        CP    E
        JR    NZ,RVSLP1
        LD    A,H
        CP    D
ENDRV1  JR    NZ,RVSLP1
        LD    A,(LASTCH)
        RET
```

e fim da palavra; mas, em vez de atuarem como índices de um endereço de base, são usados como endereços indiretos (a instrução LD A, (DE) significa "carregar o acumulador com o byte cujo endereço está em DE"). Todos os pares de registradores do Z80 podem ser usados dessa maneira. Uma limitação peculiar do conjunto de instruções é a falta de qualquer comando de comparação de 2 bytes. Assim, para comparar os conteúdos de DE e HL, é necessário comparar E com L e depois D com H. Da mesma forma, na versão para o 6502, comparam-se X e Y indiretamente, usando uma posição da memória, já que não há instrução para se comparar X com Y.

ELEBRA

Destacando-se na fabricação de superminis, a Elebra Computadores é uma empresa nacional que vem investindo no aprimoramento de seu quadro de profissionais especializados e na preparação de recém-formados.

Quando, após quinze anos de atividade, a Elebra se viu ameaçada na greve dos metalúrgicos de maio de 1985, dentro da empresa aconteceram mudanças importantes. A indústria, que já vinha investindo no desenvolvimento de seus engenheiros e técnicos, deu início a uma nova área, responsável pelo trato daquela tensão: a de recursos humanos.

Hoje ela possui uma assessoria de relações sindicais, que funciona como ponte de comunicação entre seus funcionários, o sindicato e a diretoria. Dessa forma, o relacionamento entre os três grupos tornou-se aberto, inclusive com formação de comitês de negociação.

Mas a preocupação primordial da Elebra continua sendo a formação de técnicos e engenheiros adequados ao desenvolvimento da empresa. Por essa razão existe o chamado plano de carreira, que visa a especializar profissionais em funções de criação, bem como a fixá-los em atividades indispensáveis à elaboração de projetos e inovações no setor tecnológico. Assim, os técnicos podem ter acesso a planos de desenvolvimento, antes privilégio apenas de engenheiros.

Superminicomputador
Com arquitetura de 32 bits, o sistema MX850 suporta todas as principais linguagens. Além disso, opera em tempo compartilhado e tempo real simultaneamente com processamento por lotes.

A Elebra investe ainda na formação profissional de estagiários e recém-formados. Para isso, mantém um relacionamento com as faculdades que orientam os estagiários dentro de suas especialidades. Na empresa, eles recebem treinamento nos equipamentos adequados para sua formação tecnológica.

Área técnica

A Elebra Eletrônica é um conglomerado de empresas de equipamentos para telecomunicações, computadores, periféricos e semicondutores. Acompanhando o enorme avanço tecnológico nessa área, a empresa prepara-se para atuar numa faixa ainda inexplorada do mercado: produção e comercialização de equipamentos de mecânica fina. Esses componentes são imprescindíveis na fabricação de impressoras, mas ainda não possuem similares nacionais, encarecendo, portanto, o custo do produto final.

No campo das telecomunicações, a Elebra desenvolve projetos tecnológicos como, por exemplo: equipamentos PCM de primeira, segunda e terceira geração, multiplex telegráfico digital, amplificadores de linha, sistema de tarifação e supervisão de centrais telefônicas e concentrador de dados.

Interligando as áreas da telecomunicação e da informática, a Elebra desenvolve uma linha completa de modems de alta e de baixa velocidade, além de acessórios para transmissão de dados.

Computadores e periféricos

A Elebra entrou no ramo de computadores em outubro de 1984, a partir da associação com o Bradesco e a Medidata. Mas sua imagem está intimamente ligada à Mônica, a impressora fabricada pela empresa e de grande aceitação no mercado. Esse equipamento imprime cem caracteres por segundo e o modelo E16030 pode formar colunas de até 136 caracteres.

Por um acordo firmado com a indústria americana Digital Equipment Corporation, a Elebra conseguiu acesso à tecnologia do supermini VAX 11/750. E com base nessa tecnologia a empresa lançou seu primeiro computador, o supermini MX850, destinado ao uso geral e orientado para processamento distribuído.

O MX850 utiliza o vasto software básico e aplicativo desenvolvido para o VAX, combinando uma arquitetura de 32 bits com um sistema operacional de memória virtual que pode chegar a 1 Gbyte (um gigabyte, ou seja, 1 bilhão de bytes). Seu desempenho em ambientes de multiprogramação permite atender às necessidades de processamento em tempo real, em tempo compartilhado ou em lotes (batch). Além disso, ele aceita todos os tipos de linguagem de programação.

Quanto aos periféricos, além das impressoras Mônica, a Elebra produz também discos flexíveis e rígidos. Dos flexíveis, citam-se os 9408/A, B e C, de 5 1/4 polegadas, com capacidades de 256 Kbytes, 512 Kbytes e 1 Mbyte respectivamente. Os rígidos são o CMD 9448 e o W9034. O primeiro possui cartucho removível de 16 Mbytes. O segundo é uma unidade Winchester de 9 polegadas, com capacidade de 340 Mbytes.

A Elebra também fabrica fitas magnéticas, interface serial para impressoras Mônica e a impressora Alice, com velocidade de 250 cps e 136 colunas.

ROBÔS PARA MONTAR

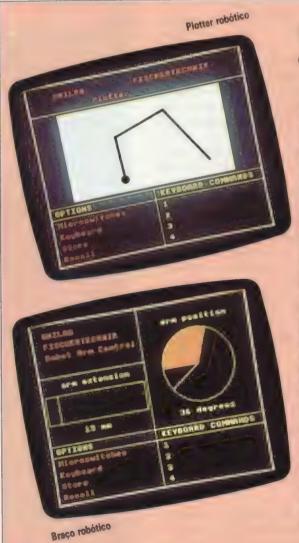
Os kits robóticos nem sempre propiciam passatempo satisfatório. Pensando nisso, a Fischertechnik produziu um pacote que permite ao usuário elaborar seus próprios projetos, além dos seis que o acompanham.

Embora profissionais da indústria de microcomputadores tenham previsto um grande desenvolvimento no campo da "robótica doméstica", ou seja, dos robôs para o lar, até o momento isso ainda não ocorreu. As razões são mais ou menos óbvias. Apesar dos inúmeros robôs domésticos existentes no mercado, todos ainda apresentam consideráveis imperfeições, deixando o público em dúvida sobre se de fato vale a pena iniciar-se na robótica.

Existem, no mercado internacional, robôs de fácil montagem, como os Movits, que pessoas sem conhecimento de robótica ou eletrônica conseguem montar em poucas horas. O problema desses dispositivos é terem aplicações extremamente limitadas, fazendo o usuário, após algum tempo, perder o interesse por eles. O mercado oferece também robôs mais caros, cuja operação requer em geral amplas noções de robótica e eletrônica. Muitos desses robôs, porém, têm uma única aplicação. Isso significa que, quando todas as possibilidades foram exploradas, o robô é posto de lado. Assim, parece não haver dúvida de que seria desejável um robô de fácil montagem e capaz de executar várias tarefas.

A isso se propôs a Fischertechnik Robotics Kit, baseando seus kits numa idéia bastante simples. Há anos encontram-se no exterior diversos conjuntos eletrônicos para montar, destinados a crianças, que possibilitam a construção de circuitos simples, como, por exemplo, um rádio elementar. Pode-se desmontar e remontar esse mesmo kit, de modo a formar um timer ou um alarme anti-roubo. Muitos de nós brincamos em criança com jogos de armar, que permitem a construção de inúmeros modelos diferentes. A Fischertechnik, ao projetar seus kits robóticos, associou essa idéia à moderna tecnologia de computadores. Como resultado, desenvolveu um produto que poderá significar para a robótica o avanço e difusão que tantos vêm anunciando.

O Kit Robotics oferece ao usuário a possibilidade de construir diferentes dispositivos robóticos acionáveis por microcomputador: desde um braço mecânico e uma máquina que ordena objetos de vários comprimentos, até um plotter e um dispositivo para entrada de gráficos. Após realizar suas experiências, o usuário tem a opção de desmontar o robô para construir outro. O controle por computador requer uma interface, para transformar os sinais digitais do computador em sinais que poderão ser usados na operação dos motores elétricos do kit.

Como as peças são de material plástico, podem-se encaixar os componentes tal como num jogo de armar comum. Blocos ligados formam as partes mais longas dos braços ou servem para sustentar as unidades de controle. Outras peças destinadas à construção das unidades mecânicas são as engrenagens e os parafusos sem fim que as fazem girar. Há também uma grande placa de plástico, medindo 260 x 187 mm, usada como base para o robô — ou para o papel, caso se incorpore um plotter ou um tablete gráfico. O controle elétrico e a alimentação também

Monitorando o movimento

Ao lado, vêem-se as telas iniciais do software da interface. A foto de baixo mostra o programa que controla o braço mecânico. À direita da tela, um diagrama circular registra o movimento horizontal. À medida que o braço deixa a posição zero e vai para a direita, a linha amarela se move no sentido horário, e o número de graus indicado embaixo aumenta. Com a extensão do braço, a linha dentro da barra (na parte esquerda do vídeo) e os milímetros mudam conforme o movimento "real". Na parte inferior da tela, estão as opções de comando disponíveis controladas por microinterruptores da placa ou pelo teclado. As sequências de movimentos armazenadas no computador podem ser chamadas e executadas de novo.

Na tela superior, temos o mesmo sistema de controle com o plotter robótico, com a diferença de que se acrescentam comandos extras para levantar e baixar a caneta sobre o papel. À medida que o plotter desliza pelo papel, os movimentos são espelhados na tela.

requerem montagem de componentes. Os kits incluem vários conectores macho e fêmea, oito interruptores e dois motores elétricos. Acompanham ainda cerca de 1 m de cabo flexível de vinte vias e um pequeno cabo suplementar, para emendas. As peças são robustas e dão impressão de grande durabilidade. A única exceção é a base do potenciômetro, que se quebra com relativa facilidade.

O potenciômetro apresenta uma pequena desvantagem: ao contrário do restante da instalação elétrica, cuja fiação pode ser parafusada nos conectores, a única maneira de se ligarem os fios desencapados ao potenciômetro é enrolando-os em volta dos conectores, e soldando-os, a fim de evitar curtos-circuitos, que danificariam o potenciômetro.

Ajuste dos componentes

A construção desses robôs não exige habilidade especial. A maior parte de suas peças é bem resistente, o suficiente para que mesmo dedos inexperientes consigam manejá-las com sucesso. O maior problema envolvido na montagem está no próprio manual — talvez a parte mais fraca do conjunto. Ao lado de aspectos positivos (como, por exemplo, as instruções para as conexões dos circuitos, bem fáceis de entender), o manual baseia-se em ilustrações pouco claras de modelos parcialmente construídos. Fotos das peças que compõem determinado modelo são mostradas junto com outras do robô em construção, mas ressentem-se de maior detalhamento, além de não virem acompanhadas de legendas. Isso faz o construtor despender um grande esforço para tentar descobrir em que lugar deve instalar determinado componente. Além disso, como o robô é mostrado apenas num ângulo, fica-se tentando adivinhar onde encaixar certa peça quando ela não aparece na ilustração.

Há um outro agravante: as fotos são em branco e preto. Então, quando se chega à etapa final, que corresponde ao ajuste dos circuitos, torna-se quase impossível visualizar o procedimento correto de instalação. Um fato estranho, considerando-se que o circuito depende em grande parte da codificação pelas cores do cabo flexível. A quantidade de informações fornecidas pelos diagramas de instalação é mínima e de pouca ajuda; simplesmente estabelecem, por exemplo, que os fios de E3 a E8 pertencem a determinada seção.

Apesar disso, devemos esclarecer que o robô mostrado na foto deste artigo foi montado por uma pessoa totalmente leiga em robótica e eletrônica. Mesmo assim, seria conveniente que o fabricante não confiasse tanto no bom-senso das pessoas e melhorasse o manual.

Mãos à obra

Montado o robô, a tarefa seguinte é conectá-lo a uma interface. A placa desta consiste em vários conectores, divididos em seções, para prender os fios condutores do cabo flexível. A inter-

Eletroímã
A barra de metal no topo do eletroímã encaixa-se na garra do braço. Pode-se ligar e desligar o ímã, possibilitando ao robô levantar pequenas placas de metal fornecidas com o kit.

Potenciômetro
Usam-se os potenciômetros para realimentar informações ao computador, que assim determina a posição correta do robô

Motor
Uma caixa de plástico abriga o motor, conectado por duas saídas. O parafuso sem fim, ao girar, impulsiona as engrenagens que movem os robôs.

Lâmpadas
Sempre que um dos três sistemas de controle principais (ímã, motores de movimentos horizontal ou vertical) é acionado, uma delas acende, informando qual controle está sendo usado

Interruptor
Há oito, que correspondem às oito linhas de dados no conector de expansões do micro. Combinados, geram um número no registrador de direção de dados; este orienta os movimentos do robô.

Força para o robô
Esta interface permite que se controle um kit robótico por intermédio de um micro. Os fios do robô encaixam-se nos conectores da interface, que, por sua vez, liga-se ao micro pelas saídas das expansões e da impressora.

Discos de metal
Os discos de metal permitem ao robô produzir trabalho útil.

Mecanismo de engrenagem
Os dois motores elétricos movimentam o robô por intermédio de mecanismos de engrenagem que, embora complicados, funcionam bem. Outros sistemas de engrenagens fornecem a realimentação para os potenciômetros.

face mostrada na foto é a do BBC Micro, embora as de outros computadores talvez não difiram muito. No canto direito da placa estão as entradas para interruptor. Sob elas estão os dois potenciômetros, que possibilitam ao computador monitorar a energia fornecida aos interruptores e às lâmpadas elétricas. A saída analógica mede o controle dos potenciômetros.

No lado contrário da placa de interface encontram-se as conexões que controlam os motores e um eletroímã (usado pelo braço mecânico para para apanhar objetos de metal). Finalmente, há o conector da fonte de energia (6 a 8 V), que alimenta todo o sistema.

A interface conecta-se ao BBC Micro por três saídas separadas (o conector de expansões, a saída analógica e a da impressora), utilizadas para controle dos motores e do eletroímã. Como o sistema se destina a usuários inexperientes, construiu-se a placa de interface prevendo-se proteção em caso de ligações incorretas, que poderiam provocar voltagens elevadas nas interfaces do computador e eventualmente danificá-lo.

A linguagem Prof

Juntamente com a interface, o kit traz um cassete com três programas que auxiliam no controle do robô. Os dois primeiros, especializados, controlam o braço e o plotter, permitindo aos dispositivos externos o controle pelo teclado ou pelos interruptores do próprio robô. O terceiro programa, o Prof (abreviação de professor), é um sistema operacional genérico, através do qual comanda-se o robô diretamente. Trata-se de uma extensão do BASIC, acrescido de alguns comandos. Muitas das palavras adicionais introduzidas pelo Prof são variáveis — como Motor e Magnet (ímã) — para os endereços de controle, enquanto outras servem para auxiliar a programação das seqüências de movimentos que os robôs deverão realizar.

O Prof, sem dúvida, é uma linguagem com muitos recursos, permitindo loops condicionais e programação estruturada, mas confina-se principalmente ao controle de saída. Assim, não dá suporte ao painel gráfico, um dispositivo de entrada, porque a linguagem não armazena informações provenientes de dispositivos de entrada. Por conseguinte, para tais fins, o usuário deverá criar seus próprios programas em BASIC.

Apesar dos problemas com a documentação, o kit e a interface da Robotics constituem um sistema razoável. Os robôs têm um desempenho satisfatório e podem ser construídos em poucas horas. O usuário não precisa se ater apenas aos seis projetos que acompanham o kit. Uma vez familiarizado com o sistema, poderá construir inúmeros modelos.

O kit não é um brinquedo barato e o manual tem os seus defeitos. Mas, uma vez superadas essas dificuldades, o kit e a interface da Fischertechnik Robotics irão firmar-se como uma iniciação ideal para o usuário interessado em explorar o campo da robótica.

GERADORES DE PROGRAMAS

O usuário já pode desenvolver seus próprios aplicativos. Com os geradores de programas, instruções genéricas em linguagem muito próxima da comum são passadas ao BASIC sem erros.

É possível um software que elimina o trabalho árduo e maçante da programação, produzindo seqüências de instruções em BASIC sem erros de sintaxe? Nem sonho nem ficção científica, os geradores de programas são ferramentas já em uso por programadores e software houses. Eles transformam instruções genéricas, emitidas pelo usuário em linguagem próxima da comum, em instruções em BASIC, com sintaxe correta e rotinas de erro incorporadas.

Essas características estão presentes, em algum grau, em linguagens como LOGO e FORTH, que não aceitam a introdução de palavras estranhas ao código ou não definidas pelo usuário. Contudo, esses recursos "inteligentes" de pouco valerão se quisermos, por exemplo, programar o cálculo de nosso imposto de renda.

Hoje, porém, a multiplicação de programas incorporando conceitos de inteligência artificial torna menos remota a possibilidade de um usuário leigo criar aplicações sob medida com um mínimo de esforço..

O Microtext, por exemplo, um software desenvolvido na Inglaterra, gera questionários, sistemas de informação e programas educacionais. Como mostra a ilustração, Microtext pode criar programas que instruem um usuário leigo a executar complexas tarefas técnicas. No modo de autoria, o Microtext permite testes e alterações; finalizando o programa, o software transforma-o num sistema pronto para execução, que o usuário não pode mais alterar. Disponível no mercado internacional, o Microtext roda em CP/M.

Outro programa bastante sofisticado é o The Quill, dedicado a gerar jogos de aventura. Ele permite montar um banco de dados de informações — lugares, objetos, como manipulá-los etc. — e incorpora todos esses dados em linguagem ASSEMBLY, produzindo um programa que roda sem erros desde a primeira execução. Um acréscimo útil é o programa Illustrator, que gera gráficos para as aventuras do The Quill.

Este e o Illustrator foram desenvolvidos para rodar em máquinas de recursos modestos, como o Sinclair Spectrum e o Commodore 64. Apesar do baixo preço, o The Quill já foi utilizado por software houses ingleses para produzir diversos jogos de aventuras lançados comercialmente.

The Quill e Microtext, contudo, não rivalizam em capacidade com os mais avançados geradores: The Last One (pretensiosamente anunciado como "o último software que você precisará comprar") e o Sycero, ainda mais poderoso, porém menos "amistoso" na interação com o usuário. No Brasil, foi desenvolvido o Gensoft I, que opera com comandos em português.

Embora ambos sejam geradores de programas, Sycero e TLO (como os distribuidores preferem agora chamar The Last One) diferem em concepção, abrangência e na maneira de definir as necessidades do usuário. Ambos geram programas em BASIC, e são extremamente eficientes na manipulação de dados, incluindo as rotinas de entrada, processamento, armazenamento e acesso a dados, assim como na impressão de relatórios de saída. Qual será, então, a diferença, entre esses geradores de programas e um SGBD (sistema gerenciador de banco de dados) convencional? É que os programas criados pelos geradores tornam-se inteiramente independentes, enquanto os bancos de dados só permitem desenvolver programas executados a partir do sistema gerador. Em relação a um avançado SGBD como o dBase III, por exemplo, podemos dizer que os geradores aqui examinados não dispõem de tantos recursos, mas são bem mais fáceis de utilizar.

Nem o Sycero nem o TLO se prestam à geração de qualquer tipo de programa. Jogos, por exemplo, estão fora de sua alçada — mesmo um simples jogo de aventura baseado apenas em textos. Tampouco podem produzir uma planilha financeira ou um processador de textos. Seu valor revela-se, contudo, na manipulação de qualquer tipo de dado, do cálculo de impostos às complexas equações de engenharia.

The Last One mostra-se mais adequado às inclinações do usuário moderno, que prefere fazer todo o pré-planejamento diretamente na tela, sem rascunhos, papel e lápis. Contudo, trabalhando dessa maneira, o usuário não estará extraindo do software todo seu potencial. O plano de ação do Sycero, que consiste em sete pontos, enfatiza a importância do planejamento prévio:

- Planejar;
- Especificar o sistema;
- Desenhar o lay-out das telas;
- Verificar os dados;
- Definir o programa;
- Produzir relatórios;
- Gerar o programa.

Aliás, os manuais de ambos os produtos realçam o seguinte ponto: mesmo utilizando ferramentas que eliminam a parte mais mecânica da programação, o usuário continua responsável pela cuidadosa organização e planejamento prévio de seu programa.

"A correção de erros por mau planejamento às vezes demora mais que o próprio desenvolvimento do sistema. Fuja da tentação de pensar num problema durante 5 minutos e, em segui-

da, correr para colocar na máquina o disco do Sycero, construindo o sistema enquanto trabalha. Esse não é um bom método.

"Comece sempre fazendo uma lista detalhada de tudo o que você pretende que o computador produza, tanto dados mostrados na tela (informações on line) como relatórios escritos. Só depois de determinar com precisão o que deseja é que deve definir quais as informações que entrarão, a fim de obter tais resultados."

O manual do TLO também dá conselhos objetivos:

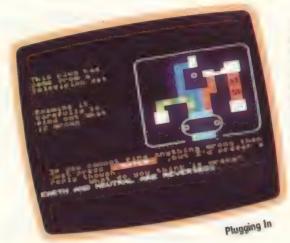
"Planeje o fluxo geral de seu programa e antecipe os erros que o usuário final irá cometer; prepare o programa, então, para deter esses erros com segurança. Na verdade, será melhor planejar tão detalhadamente quanto você faria com um programa comum. Para tanto, inicie elaborando diversos programas curtos, interligados por um simples menu, que chama cada um quando necessário. Você verá que esse sistema de construção modular permite criar com facilidade programas extensos e complexos".

Qualquer software que objetive possibilitar ao usuário novato a criação de suas próprias aplicações computacionais precisa satisfazer plenamente alguns quesitos. Portanto, deve:

- Encorajar a programação estruturada "de cima para baixo", na qual o usuário primeiro esboça os contornos gerais do programa e, depois, especifica cada ponto com mais detalhes, conforme dá seguimento ao trabalho;
- Ser dirigido por menus de múltipla escolha, com mensagens que orientem o usuário na escolha das opções disponíveis;
- Detectar erros assim que são introduzidos, e permitir correções e melhoramentos;
- Ser "transportável", isto é, capaz de rodar em outro equipamento, independente do sistema de software que o gerou;
- Produzir ampla documentação do processo de geração do programa, de modo que um novo usuário, mesmo desconhecendo as etapas de criação, possa introduzir melhorias ou correções.

Todas essas medidas auxiliam sobretudo o usuário novato. Contudo, os geradores de programas não são ferramentas restritas aos iniciantes; devem ser aceitáveis também para os mais experientes, não os enredando e atrasando com excesso de testes e sistemas de segurança.

Tanto o Sycero como o TLO operam por meio de menus informativos. Dos dois, o segundo é o que mais segue a abordagem estruturada. Ele começa produzindo um fluxograma, na verdade uma série de comentários esclarecedores sobre a finalidade de cada seção do programa. Por exemplo, Desvio para menu de quatro opções: esse comentário, se se desejar, pode ser incorporado ao programa final em BASIC sob a forma de linhas de REM. Definidas as entradas e as telas, o programa resultante em formato ASCII deverá ser gravado em código binário.

Plugging In

Este programa instrucional gerado pelo Microtext apresenta questões a respeito de sistemas elétricos. Aplicações educacionais, que envolvem interação com o usuário e comparação de suas respostas com dados pré-gravados, são ideais para os geradores de programas.

Já no Sycero, começa-se por definir os arquivos e o desenho das telas. Em seguida, vem o processamento das variáveis de entrada, ou seja, a associação de determinados dados de entrada às respectivas mensagens de erro ou de auxílio ao usuário. Definem-se também os relatórios de saída e outras opções. Os vários módulos são então interligados, imediatamente antes de serem codificados em linhas de programa.

Embora esse processo não seja tão fácil para o novato, tem a vantagem de exigir uma cuidadosa definição do sistema todo antes da fase de programação; caso contrário, o programa simplesmente não roda. Se forem detectados erros durante a codificação, o gerador interrompe o processo e informa qual o erro em questão, possibilitando ao usuário corrigi-lo. Ao fim da codificação, o Sycero automaticamente grava o programa num arquivo binário, que pode ser executado a partir do próprio gerador.

Tanto o Sycero como o TLO incluem complexas e abrangentes rotinas de detecção de erros nos programas que geram, aumentando consideravelmente seu tamanho em relação a um programa equivalente em BASIC.

Versão brasileira

No Brasil, a Gensoft Informática Ltda., com sede em São Paulo, desenvolveu o Gensoft I. Segundo os fabricantes, o Gensoft I alia um gerador de aplicativos a um gerenciador de banco de dados relacional. Isso permite que, com pouquíssimos comandos — todos em português —, o usuário crie seus programas, definidos de forma coloquial, sem necessidade de conhecimentos específicos de informática ou de linguagens computacionais. Basta definir, na tela, os relatórios de que se necessita, as informações a serem armazenadas e processadas, e determinar as relações, os testes e os cálculos com os dados dos arquivos. A rapidez de execução do Gensoft I é garantida por suas rotinas em ASSEMBLY. Compõe-se de oito módulos, sendo três de desenvolvimento (texto, formato e cálculo) e cinco de execução (arquivos, utilitários, tratamento, relatórios e menu do usuário). O próprio

Faça você mesmo
The Last One (TLO) e Sycero são softwares que permitem ao usuário criar programas em BASIC bem estruturados e livres de erros de sintaxe, e que são executados independentemente do software gerador. Contudo, para utilizá-los com proveito é necessário um cuidadoso planejamento.

software se encarrega de gerar também a documentação do sistema, que tem amplos limites quanto à quantidade e tamanho máximo de arquivos, registros e campos.

No que diz respeito à documentação, tanto o Sycero como o TLO nada deixam a desejar, gerando listas de características tais como nomes das variáveis utilizadas, formatos de tela etc. O Sycero acrescenta automaticamente a data e a hora, tanto em tela como em cópias impressas, de modo a facilitar a distinção entre versões antigas e mais recentes de um programa.

Etapas da criação

Quais as opções principais do TLO? Inicialmente se oferece ao usuário um menu de oito opções.

Criar programa	Consultas
Modificar programa	Formatar novo disco
Modificar arquivo	Continuar a codificação
Definir arquivo	Voltar ao BASIC

Depois de escolher a primeira opção e responder afirmativamente à pergunta Seu programa requer arquivos?, definem-se os arquivos necessários. Assim como o tamanho e o tipo dos campos. A tela apresentará então o Menu de criação e de fluxograma. Em sua última versão, o TLO apresentava vinte opções nesse ponto:

Listar fluxograma	Reinicializar
Modificar fluxograma	Posicionar ponteiros
Codificar programa	Ler arquivo
Unir fluxogramas	Gravar em arquivo
Interromper	Pesquisar ou classificar arquivo
Entrada pelo teclado	Unir programas
Exibir dados	Verificar registros
Desvios	Limpar arquivo
Cálculos	Funções de banco de dados
Funções especiais	Multifunções

Utilizando essas opções, elabora-se um fluxograma, a exemplo deste a seguir, que atualiza e pesquisa um arquivo de endereços:

1 ..Desvio para um menu de três opções
2 ..Posicionar ponteiro no fim do arquivo
3 ..Entrada pelo teclado
4 ..Gravar dados de entrada no arquivo
5 ..Perguntar <Terminou?>.Desviar se "Não"
6 ..Desvio incondicional direto
7 ..Desvio para um menu de três opções
8 ..Posicionar ponteiro no início do arquivo
9 ..Pesquisa do arquivo pelo teclado
10 ..Exibir dados do arquivo
11 ..Perguntar <Outra consulta?>.Desviar se "Sim"
12 ..Desvio incondicional direto
13 ..Classificar arquivo
14 ..Posicionar ponteiro no início do arquivo
15 ..Ler dados do arquivo
16 ..Exibir dados do arquivo
17 ..Desvio incondicional direto
18 ..Finalizar

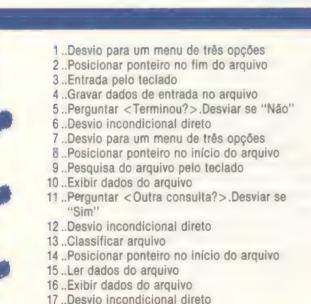
Definida assim a estrutura do programa, o gerador pode iniciar a codificação. Até aqui, não se pode sair do TLO sem perder todo o trabalho feito. Iniciada a codificação, contudo, é possível interrompê-la a qualquer ponto, e reiniciá-la por meio da função do menu inicial Continuar a codificação. Nessa fase, definem-se o desenho das telas e os destinos dos desvios; no exemplo dado, o menu definido no ponto 1 irá desviar para 2 (gravar novos dados), 7 (pesquisar os dados do arquivo) ou 18 (finalizar).

Convém gravar em separado as telas definidas; o Sycero faz isso automaticamente, ao passo que, no TLO, requer-se um comando especial.

O menu de abertura do Sycero oferece treze opções, que serão utilizadas mais ou menos na ordem em que vêm listadas:

Configuração do sistema
Inicialização
Definição dos arquivos e dos campos
Definição das telas
Processamento das telas
Definição de relatórios
Processamento dos relatórios
Definição do programa
Geração do programa
Criação de arquivo de dados reais
Execução do programa gerado
Utilitários (programas de apoio)
Finalizar a sessão

Seguindo esses passos, ao atingir a etapa Geração do programa, a maior parte do trabalho já estará pronta. As fases que se seguem requerem poucas intervenções do usuário, a não ser que o software detecte um erro básico. Em outras palavras, quanto mais detalhadas forem as fases iniciais de planejamento, mais rápida e simples será a criação do programa.

Os geradores de programas se assemelham aos compiladores, que transformam as instruções no BASIC ou outra linguagem de alto nível em comandos de código de máquina. Em ambos os casos, o software é uma ferramenta que traduz comandos genéricos, mais próximos ao usuário, para códigos mais próximos da máquina. Os geradores podem fazer essa conversão, que é um trabalho mecânico, mas não deduzem as intenções do usuário. O principal, portanto, consiste em instruir o sistema com a máxima precisão.

MÚLTIPLA ESCOLHA

Examinamos aqui os vários métodos de efetuar a subtração e a multiplicação em linguagem ASSEMBLY. Apresentamos também um novo tipo de operadores lógicos — os de deslocamento e rotação.

Tanto o Z80 como o 6502 admitem a instrução SBC ("subtração com transporte"), mas sua execução difere bastante em cada processador. O 6502 usa o flag de transporte para lidar com o recurso "empréstimo", que equivale ao transporte na adição. Na linguagem ASSEMBLY do Z80, a instrução SBC funciona exatamente como a ADC — o flag de transporte indica o resultado da operação.

Vamos supor que acrescentamos $E4 a $5F, usando ADC, tendo antes zerado o flag de transporte. O resultado no acumulador é $43, com o flag de transporte ativado, mostrando que o resultado verdadeiro é $0143. Houve um estouro para o flag de transporte, porque o tamanho do acumulador não basta para conter o resultado total.

Suponhamos, agora, ainda no Z80, que zeramos mais uma vez o flag de transporte e subtraímos $E4 de $5F: o resultado no acumulador será $7B e o transporte será ativado. Agora, se acrescentamos $7B a $E4 (tendo zerado mais uma vez o flag de transporte) verificaremos que o resultado no acumulador é $5F e que o flag de transporte encontra-se ativado. Isso está totalmente coerente, como demonstra

$5F – $E4 = $7B	transporte ativado
$5F = $E4 + $7B	transporte ativado

Se tomamos o estado do flag de transporte com indicação de que ocorreu um resultado negativo, poderemos interpretar $7B como um complemento de 2:

$7B em binário	= 01111011
Subtraindo 1	–1
Dá o complemento de 1	01111010
Negação	
Dá o complemento de 2	10000101 = 85

Deveríamos esperar então que $5F – $E4 resultasse no número negativo –$85. Verifiquemos esse resultado em notação decimal:

$5F =	95 em decimal
–$E4 =	288 em decimal
$85 =	–133 em decimal

Até aqui, portanto, tudo isso faz sentido. Suponhamos agora que a subtração em questão tenha sido uma soma de 2 bytes: $375F – $21E4. Vamos efetuá-la byte a byte:

Superior	Inferior	
$37	5F	= 14.175 em decimal
–$21	E4	= –8.676 em decimal
$15	7B	= 5.499 em decimal

Efetuando a subtração do byte inferior em primeiro lugar, o resultado será $7B e um transporte. Soma-se então esse transporte a $21, pela ins-

Deslocamentos na multiplicação
Este exemplo mostra uma multiplicação de 4 bits para maior clareza, pois o número de bits não afeta o algoritmo. Vê-se que o produto se forma pela adição de 0 ou de versões deslocadas do multiplicando, dependendo do valor de cada bit do multiplicador: 0 ou 1. Os bits do multiplicador deslocam-se para a direita através do flag de transporte, enquanto os bits do multiplicando deslocam-se para a esquerda, a partir do byte inferior até o byte superior, através do flag de transporte.

Após um deslocamento: Superior	Após um deslocamento: Inferior	Após dois deslocamentos: Superior	Após dois deslocamentos: Inferior	Após três deslocamentos: Superior	Após três deslocamentos: Inferior	
	0001		0000		0000	MULTIPLICADOR
0001	0010	0010	0100	0100	1000	MULTIPLICANDO
0000	1001	0001	1011	0001	1011	PRODUTO

trução SBC, resultando $22, que é então subtraído de $37, dando $15. A resposta, $157B, está correta, como prova a versão em decimal.

Como vemos, a aritmética de 2 bytes no Z80 segue estas regras simples:

1) Zerar o flag de transporte;
2) Subtrair os bytes inferiores, com transporte;
3) Subtrair os bytes superiores, com transporte.

A versão dessa seqüência no 6502 difere quanto ao primeiro aspecto — deve-se ativar, e não zerar, o flag de transporte, para que os bytes inferiores "emprestem" do byte superior. Não havendo empréstimo, a subtração prossegue normalmente, e o flag de transporte permanece ativado para subtração dos bytes superiores, o que ocorre de modo semelhante. Entretanto, se ocorrer um estouro negativo na subtração do byte inferior, o flag de transporte atuará como o "nono bit" do acumulador. Isso assegura um resultado correto e a desativação do flag de transporte. Quando se subtraem os bytes superiores com o flag de transporte desativado, o efeito é o mesmo que na subtração do byte superior no Z80, com o flag de transporte ativado — decrementa-se o número a ser subtraído antes que a operação ocorra. Ambas as maneiras de lidar com o empréstimo na subtração equivalem aos antigos métodos aritméticos de "emprestar" daqui e "devolver" ali. Examinemos mais detalhadamente a versão do 6502.

Se zerarmos o flag de transporte e subtrairmos $E4 de $5F, o resultado no acumulador será $7A e o flag de transporte permanecerá em zero. Vimos, pelo exemplo do Z80, que o resultado "verdadeiro" é $7B, com o flag de transporte indicando um número negativo. O valor $7B é o complemento de 2 da resposta "real" (−$85). Vemos que $7A é o complemento de 1 desse número e que o flag de transporte funciona como um interruptor para o modo do acumulador: ele é ativado para o complemento de 2, e zerado para o complemento de 1.

Se efetuarmos a subtração no 6502 com o flag de transporte ativado, o acumulador conterá $7B, e o flag de transporte será zerado. Se a subtração for de 2 bytes, zerar o flag de transporte vai decrementar o resultado da subtração do byte superior, compensando assim o "empréstimo" dos bytes inferiores.

Operação de deslocamento
Usam-se as instruções de deslocamento e de rotação para examinar o conteúdo de um registrador, bit a bit. A cada deslocamento, o bit superior ou o inferior do registrador é deslocado para o flag de transporte do PSR; o estado desse flag serve para que uma instrução de desvio dirija o fluxo de controle do programa. As instruções de rotação agem de modo a preservar o conteúdo do registrador, mas as instruções de deslocamento lógico colocam 0 no registrador à medida que deslocam bits para fora. Desse modo, um deslocamento para a esquerda multiplica por dois o conteúdo do registrador, e um deslocamento para a direita divide-o por dois.

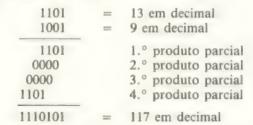
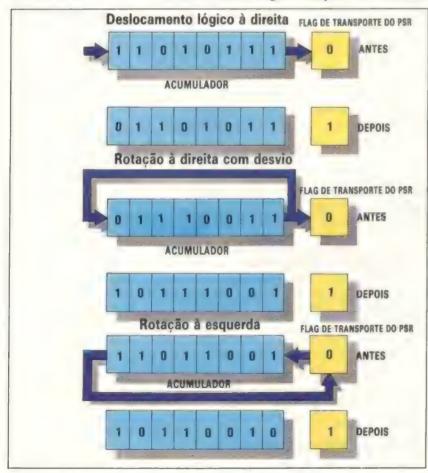
Multiplicação

Vejamos como se efetua a soma na multiplicação decimal:

174	multiplicando
x209	multiplicador
1566	1.° produto parcial
000	2.° produto parcial
+348	3.° produto parcial
36366	produto final

Não é necessário entender a notação posicional para utilizar esse método; basta seguir procedimentos simples, efetuando multiplicações de um único dígito. A idéia básica nesse método é colocar cada produto parcial numa posição à esquerda do produto anterior. Uma vez admitida a necessidade desse posicionamento, a formação dos produtos parciais exige apenas o conhecimento da tabuada de multiplicação.

A técnica de deslocamento de produtos parciais e a memorização da tabuada é o que dificulta, para muitos, o aprendizado das multiplicações decimais longas. Já na multiplicação binária há apenas um produto real, que é um multiplicado por 1; todos os outros produtos de um só dígito resultam em 0. Observe esta longa multiplicação binária:

1101	=	13 em decimal
1001	=	9 em decimal
1101		1.° produto parcial
0000		2.° produto parcial
0000		3.° produto parcial
1101		4.° produto parcial
1110101	=	117 em decimal

Vemos claramente nesse exemplo o deslocamento dos produtos parciais, assim como a simplicidade da multiplicação em notação binária. Cada produto parcial ou é igual a 0 ou ao multiplicando deslocado, dependendo de o bit multiplicador valer 0 ou 1. Para executar uma multiplicação binária, basta, pois, examinar cada bit do multiplicador e acrescentar 0 ao total (se o bit

for 0) ou o multiplicando deslocado (se o bit for 1). Como, então, devemos proceder na linguagem ASSEMBLY para examinar cada bit do multiplicador e para deslocar o multiplicando?

Tanto no Z80 como no 6502, testamos o estado de um bit específico num byte por meio da instrução BIT. No Z80, essa instrução toma como operandos um endereço e um número de bit, ativando o flag zero se esse bit for 0 e desativando-o se for 1. No 6502, o operando é um endereço. O conteúdo desse endereço combina-se logicamente em E com o acumulador, e o flag zero é ativado ou desativado, dependendo de o resultado ser falso ou verdadeiro. Nenhum dos métodos á adequado aqui. Seria bem mais conveniente se o bit em questão funcionasse como flag de transporte ou flag zero, de modo que o fluxo do programa desviasse automaticamente de acordo com o estado de cada bit. Ambos os processadores realizam isso por instruções shift (deslocar), que também resolvem o problema de deslocar o multiplicando.

Há uma série de instruções de deslocamento e também de rotação em ambos os processadores, embora as do Z80 sejam mais complexas que as do 6502. De modo geral, elas deslocam cada bit de um registrador uma posição para a direita ou para a esquerda. Elas diferem quanto ao tratamento dos bits das extremidades do registrador, onde um bit é deslocado para fora, numa extremidade, enquanto outro vai para dentro, na outra extremidade. Se o bit 7 for deslocado para fora do registrador e logo recolocado no bit 0, a operação será de rotação para a esquerda. Se for o contrário, o 0 deslocado para o 7, a operação consistirá numa rotação para a direita. Num caso desses, o conteúdo do registrador muda de ordem, mas não se introduzem valores novos; e, após oito dessas rotações, o registrador volta ao estado original.

Não se empregando a rotação, será necessário um ponto de chegada para o bit deslocado para fora e outro para o bit deslocado para dentro. Ambos são supridos pelos vários flags de condição do PSR (Processor Status Register, "registrador de estado do processador") e, em especial, pelo flag de transporte. Ao criar uma sub-rotina para multiplicar dois números de um só byte, precisamos deslocar o multiplicando para a esquerda e o multiplicador para a direita. Os bits deste devem deslocar-se para fora, indo para o byte superior do multiplicando, enquanto se deslocam zeros para os bits não ocupados. Os bits multiplicadores devem deslocar-se através de um flag do PSR para teste, mas seu ponto de chegada e o estado dos que se deslocaram para dentro não é importante, a menos que seja necessário preservar o conteúdo do multiplicador. Com relação ao multiplicador, interessa-nos apenas saber se o bit que sai é 1 ou 0.

Assim, dado que o multiplicador armazena-se no endereço MPR, o multiplicando em MPDLO e o produto em PRODLO e PRODHI, podemos desenvolver assim as rotinas para multiplicação:

MULTIPLICAÇÃO DE 8 BITS

6502			Z80		
	ORG	$C100		ORG	$D000
START	LDA	#$00	START	LD	BC,(MPR)
	STA	PRODLO		LD	B,$08
	STA	PRODHI		LD	DE,(MPDLO)
	STA	MPDHI		LD	D,$00
	LDX	#8		LD	HL,$00
	CLC		LOOP0	SRL	C
LOOP0	ROR	MPR		JR	NC,CONT0
	BCC	CONT0		CALL	ADDIT
	JSR	ADDIT	CONT0	SLA	E
CONT0	ASL	MPDLO	ENDLP0	DJNZ	LOOP0
	ROL	MPDHI		LD	PRODLO
	DEX			RTS	
ENDLP0	BNE	LOOP0	MPR	DB	$E2
	RTS		MPDLO	DB	$7A
MPR	DB	$E2	MPDHI	DB	$00
MPDLO	DB	$7A	PRODLO	DW	$0000
MPDHI	DB	$00	ADDIT	ADD	HL,DE
PRODLO	DB	$00		RET	
PRODHI	DB	$00			
ADDIT	CLC				
	LDA	PRODLO			
	ADC	MPDLO			
	STA	PRODLO			
	LDA	PRODHI			
	ADC	MPDHI			
	STA	PRODHI			
	RTS				

Como mostra esse exemplo, a programação no Z80 é muito simplificada devido a seus registros de 16 bits e respectivas instruções. Em especial, compare a sub-rotina ADDIT nos dois programas. Na versão 6502, ROR gira o multiplicador para a direita, através do transporte; e ASL e ROL deslocam o multiplicando em direção à esquerda, para fora de MPDLO e para dentro de MPDHI, também através do transporte. O loop é controlado pelo registrador x agindo como contador.

Na versão do Z80, SRL desloca o multiplicador para a direita, através do transporte; e SLA e RI, o multiplicando para a esquerda, em DE, através do transporte. O loop é controlado pelo registrador B como um contador. Observe que a instrução ADD admite a aritmética de 16 bits, e não sofre influência do flag de transporte — ao contrário do que ocorre com ADC.

No próximo artigo, analisaremos os métodos de divisão e examinaremos vários métodos de controlar a apresentação na tela. Isso completará nosso curso.

Exercícios

A) Desenvolver uma sub-rotina de multiplicação usando um multiplicando de 16 bits e um multiplicador de 8 a sua escolha.

B) A multiplicação não passa de uma adição repetida: desenvolva uma sub-rotina de multiplicação de 8 x 8 bits, que não utilize as instruções de deslocamento nem de rotação.

TANDY MODEL 100

Quem viaja a negócios precisa de um computador portátil. Vamos ver como trabalham três desses pesos-pena, essencialmente semelhantes, com destaque para o Tandy Model 100.

Um fabricante compra um produto acabado, muda a marca, a embalagem, alguns elementos, e vende-o como exclusivo. Esse processo é conhecido como OEM (Original Equipment Manufacture, "engenharia de griffe"). Técnica há muito aplicada em produtos como televisores e equipamentos de alta fidelidade, agora vem sendo usada no mercado de computadores. Três populares micros portáteis — o Tandy Model 100, o NEC PC-8201A e o Olivetti M10 — resultaram de um acordo desse tipo. Todos são fabricados pela mesma companhia, a indústria japonesa Kyocera, e vendidos para aquelas três empresas, que os acondicionam e comercializam com suas próprias marcas. Examinaremos em mais detalhe o Tandy Model 100 e ressaltaremos as diferenças entre ele e seus dois irmãos.

Pesando pouco menos de 1,8 kg, os três modelos sem dúvida classificam-se como portáteis. O Model 100 tem um teclado completo, tipo QWERTY, software residente em ROM incorporado e um LCD (visor de cristal líquido) acionado a bateria. O conteúdo da memória não se perde quando se desliga a máquina. Os arquivos ficam armazenados em RAM e são acessados diretamente, tal como ocorre em fita cassete ou disco. Preferindo-se armazenamento externo, o Model 100 também se conecta a um gravador cassete ou a uma unidade de disco, mas a memória permanente facilita a entrada de dados importantes quando se está em viagem.

O LCD apresenta oito linhas de quarenta caracteres, mesclando textos e desenhos. A tela compõe-se de 15.360 pontos, endereçáveis um a um. Formam-se os caracteres, maiúsculos ou minúsculos, em matrizes de 6 x 8. O Model 100 dispõe de um conjunto completo de caracteres para diversas línguas, bem como de um conjunto especial de caracteres gráficos; já a máquina da NEC possui apenas três caracteres gráficos. Os LDCs do Tandy e do NEC são fixos; já o do Olivetti pode ser inclinado, proporcionando melhor ajuste ao ângulo de visão do operador. Para compensar a ausência desse recurso, os visores dos micros da NEC e da Tandy têm controles de contraste para melhorar a visibilidade.

O teclado do Tandy é de alta qualidade e apresenta teclas especiais para acessar os elementos gráficos incorporados. Outro recurso consiste na

Saída para impressora paralela

Interface para cassete

Leitora de código de barras

CPU
Um chip CMOS 80C85 de 8 bits, de baixo consumo de energia.

Expansão da RAM
Aqui se instalam chips adicionais, que expandem a RAM do Tandy até 32 Kbytes

NEC PC-8201A
Este modelo tem as mesmas dimensões dos concorrentes mas o lay-out de seu teclado é bem diferente. As teclas do cursor agrupam-se num pequeno tablete, e as de função reduzem-se de 8 para 5. Além disso, o NEC vem equipado com apenas três programas ROM: Text, Schedule e Telecom.

Saída para modem
Saída serial padrão RS232. O software para comunicações é residente na máquina.

Fonte de alimentação
O Model 100 funciona até 20 horas com quatro pilhas alcalinas 'AA' (tipo lapiseira). A memória interna mantém-se por até trinta dias, por meio de baterias de níquel-cádmio, recarregadas automaticamente quando se liga a máquina.

ROM padrão
Este chip contém o BASIC da Microsoft e os programas residentes.

Saída para o LCD
Um cabo multivias conecta o visor de cristal líquido à placa do sistema.

Bus para o sistema e conexões para ROM adicional
Slots vagos para expansão futura da ROM do sistema e para controle de entrada e saída

RAM padrão de 8 k Kbytes

Unidade de teclado
Estes chips, que contêm os conjuntos de caracteres, controlam a entrada e a saída do teclado.

Olivetti M10
A versão Olivetti vem equipada com cinco programas residentes em ROM, além de apresentar um útil detalhe exclusivo: o visor de cristal líquido poder ser inclinado cerca de 40 graus, melhorando a visibilidade da tela sob diferentes condições de iluminação. O teclado é basicamente igual ao do Tandy.

TANDY MODEL 100

DIMENSÕES
300 x 215 x 50 mm.

CPU
CMOS 80C85 de 8 bits.

CLOCK
2,4 MHz.

MEMÓRIA
8 Kbytes ou 24 Kbytes de RAM, expansível em módulos de 8 Kbytes até um total de 32 Kbytes; ROM de 32 Kbytes, incluindo software e BASIC da Microsoft.

VÍDEO
LDC de 40 colunas x 8 linhas; gráficos de 240 x 64 pixels endereçáveis; caracteres ASCII e internacionais; 39 caracteres gráficos.

INTERFACES
Impressora paralela, fita cassete, saída serial RS232, leitora de código de barras, bus para o sistema.

LINGUAGENS DISPONÍVEIS
BASIC da Microsoft.

TECLADO
Tipo QWERTY, com 56 teclas; bloco numérico incorporado; oito teclas de função programáveis; quatro de comando e quatro de movimento do cursor.

DOCUMENTAÇÃO
Guia de referência rápida do BASIC, com 48 páginas; detalhado manual de operação, com duzentas páginas.

DIFERENÇAS
Olivetti M10:
57 teclas; 47 caracteres gráficos; tela inclinável; manual do usuário.
NEC PC-8201A:
57 teclas, sendo cinco de função; as do cursor vêm agrupadas; RAM de 16 Kbytes, expansível para 96 Kbytes; apenas três caracteres gráficos.

criação de um teclado numérico pela transformação das letras [M], [J], [K], [L], [U], [I] e [O] nos números de [0] a [6]; as teclas [7], [8] e [9] mantêm suas funções normais. Para a movimentação do cursor, os modelos Tandy e Olivetti têm quatro teclas pequenas, colocadas lado a lado, acima e à direita do teclado; no NEC, acham-se dispostas de modo mais funcional, formando um quadrado.

As três máquinas apresentam teclas de função programáveis, usadas com o software incorporado para manipulação de arquivos e dos programas mantidos em ROM. Novamente há diferenças: o Tandy Model 100 tem oito teclas de função, mais quatro adicionais para tarefas internas. Usa-se a tecla [Paste] (colar) para transferir dados de um programa para outro; [Label] para atribuir nomes específicos a cada tecla de função; [Print] para enviar arquivos para a impressora; e [Break] para interromper a execução de um programa. Essa configuração é idêntica no Olivetti, mas o NEC tem cinco teclas programáveis para até dez funções e também uma tecla [Pause].

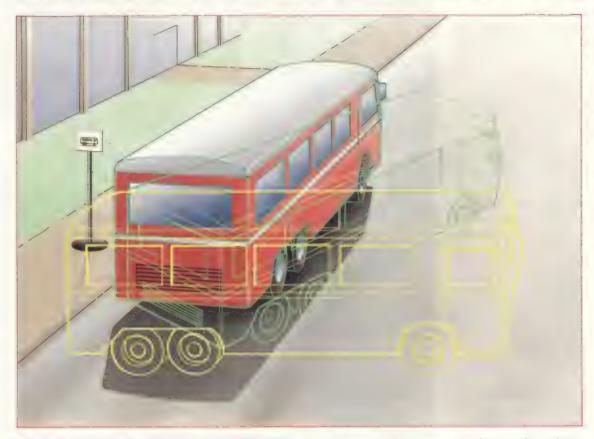
Capacidade de memória

O Model 100 e o M10 são fornecidos em duas versões: com 8 ou 24 Kbytes de RAM, expansíveis para 32 Kbytes com a adição de um módulo interno de RAM. O NEC tem maior capacidade: é apresentado com 16 Kbytes, expansíveis para 64 Kbytes internamente, ou a 96 Kbytes, acoplados ao conector incorporado.

O Model 100 vem com o BASIC da Microsoft e um pequeno sistema de gerenciamento do software interno. Ao ser ligada, a máquina lista na tela arquivos armazenados na memória, bem como os nomes dos programas residentes fornecidos com o equipamento. Um deles é o Text, um pequeno processador de textos, adequado para escrever cartas, memorandos ou pequenos relatórios; especialmente apropriado para anotações rápidas, devendo ser de grande valia para jornalistas, estudantes ou homens de negócios. Há um programa de agenda, o Schedule, um pequeno banco de dados que armazena compromissos, gastos, "coisas a fazer" e outros lembretes; uma função de pesquisa possibilita encontrar as informações rapidamente. Um terceiro programa, o Address, é um pequeno banco de dados que parece desnecessário, duplicando as funções do Schedule. Finalmente, um programa de comunicações chamado Telecom controla a saída RS232 e permite a conexão do Model 100 a um modem e a linhas telefônicas — pressionando-se algumas teclas, enviam-se ou recebem-se dados de computadores distantes. O micro da NEC vem equipado apenas com BASIC, o Text e o Telecom.

As três máquinas são bem providas de interfaces: têm saída para comunicações RS232, saída para impressora paralela, interface para gravador cassete e entrada para leitora de código de barras. Os modelos Tandy e Olivetti incluem um bus de sistema, enquanto o NEC tem duas saídas seriais adicionais.

Com pequenas diferenças entre os três modelos, a utilização de uma máquina básica significa que os fabricantes podem fornecer um produto de alta qualidade, sem que uma só companhia tenha de arcar com todo o custo de desenvolvimento. A um preço relativamente acessível para as versões básicas, esses três portáteis oferecem um bom desempenho.

RECONHECIMENTO VISUAL

Uma boa adequação
Alguns sistemas de reconhecimento do padrão adotam abordagem de cima para baixo, na qual uma área ou cena é examinada à procura de determinado objeto. Uma representação deste, segundo as linhas que delineiam seu contorno (representação "em arame"), é oferecida numa tela, em vários ângulos, até que, se o objeto estiver presente, seja encontrada a projeção que se ajusta ao esboço da figura real. Na cena tridimensional apresentada ao lado, são inúmeras as projeções possíveis "em arame" antes de se encontrar a correta.

Ver e compreender o que se passa no mundo real é uma das principais metas da inteligência artificial. E já existem sistemas capazes de reconhecer padrões — e de mudar seu desempenho conforme o que "vêem".

Como se costuma dizer, "A beleza está no olho do observador"; mas também se poderia afirmar que a beleza está no cérebro de quem a percebe — mais precisamente, numa complexa cadeia de processos neurais que começa no fundo da retina em alguma parte do córtex occipital do cérebro. É essa cadeia de eventos que os fisiologistas tentam entender e, até certo ponto, os especialistas em robôs procuram duplicar.

A percepção visual é um componente a tal ponto importante para nossa captação do ambiente que, com freqüência, "ver" é usado como sinônimo de perceber, compreender. De fato, a compreensão constitui a chave ao desafio da visão computadorizada. De algum modo, o computador deve obter informações, fornecidas por uma câmara ou outro dispositivo fotossensível, e compreender o que elas lhe dizem sobre o ambiente. Adquirir a informação é fácil, o problema está em como interpretá-la.

O processo de transformação de imagens em significados envolve três estágios principais: processamento da figura, reconhecimento do padrão e compreensão da imagem.

Processamento da figura. Confere nitidez a uma imagem borrada ou distorcida. (Problema relativamente fácil de resolver.)

Reconhecimento do padrão. Detecta a presença ou ausência de características ou objetos significativos. (Essa é uma tarefa de solução mais difícil.)

Compreensão da imagem. Entende o que se passa no mundo real. (Um problema ultracomplexo.)

O terceiro — a verdadeira visão computadorizada — ainda não foi alcançado. Contudo, já se conseguiram resultados úteis a partir dos estágios iniciais.

Reconhecimento do padrão

Vamos "ver" (entender) como funciona o segundo estágio. Em síntese, os reconhecedores de padrão classificam as imagens por comparação a um conjunto limitado de alternativas — por exemplo, as letras do alfabeto, no caso de sistemas envolvendo caracteres ópticos. Para isso, examinam partes da imagem digitalizada: pequenos grupos de pixels, os pontos e os grupos de pontos que compõem a imagem. O sistema pode então reconhecer e classificar padrões com base na presença ou ausência de certas características diferenciáveis.

Até agora foram desenvolvidas duas abordagens contrastantes da visão computadorizada na compreensão da imagem: a abordagem de cima para baixo (ou dirigida por modelo) e a abordagem de baixo para cima (ou dirigida por dados).

A análise de cima para baixo também é chamada de alucinação controlada. O sistema possui um modelo interno do que está procurando (um ônibus, por exemplo), e projeta esse modelo em vários ângulos sobre o plano da figura. A seguir, testa a adequação entre o que foi "imaginado" e o que está realmente lá.

Os sistemas de baixo para cima esquadrinham os dados da imagem, procurando linhas, margens e outros sinais reveladores. Desse modo, elaboram uma descrição simplificada e bastante abstrata da imagem, denominada esboço primário. A idéia é que, despojando-o de detalhes, será mais fácil comparar o esboço com exemplos de uma dada categoria de objetos armazenados como padrões de referência.

Sistemas práticos tendem a empregar os dois métodos, embora, até recentemente, houvesse bem poucos sistemas de aprendizagem no campo da visão computadorizada — e todos dependentes de inteligência pré-programada. Notável exceção é o Wisard (Wilkie, Aleksander & Stonham's Recognition Device, "dispositivo de reconhecimento de Wilkie, Aleksander e Stonham"). Esse equipamento pode ser programado para estabelecer distinção envolvendo sutilezas como, por exemplo, entre um rosto humano sorridente e outro carrancudo. Pode até mesmo generalizar esse conhecimento estendendo-se a rostos nunca antes "vistos". O Wisard opera em tempo real (25 quadros por segundo); seu sucesso comercial na classificação de chocolates em esteiras transportadoras pode abrir caminho para uma nova geração de "máquinas que enxergam", capazes de adaptar e aperfeiçoar seu desempenho.

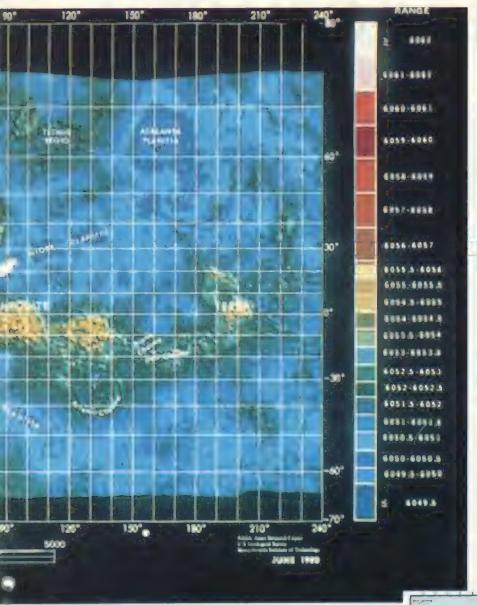
Em linhas gerais, o Wisard trabalha alocando grupos de 8 pixels de cada vez (óctuplos) em bancos de RAM. Um óctuplo é um detector de características e pode estar em um de 256 estados (de 0 a 255), dependendo do estado de cada um dos pixels que ele monitora. Durante a fase de treinamento, um 1 é armazenado na localização de memória especificada pelo estado do óctuplo quando determinada imagem está presente. Mais tarde, na fase de reconhecimento, a presença de um 1 naquela localização evidencia que a imagem "ensinada" está presente mais uma vez.

Usando-se grande número de óctuplos, ou discriminadores, o sistema se torna relativamente impenetrável a dados irrelevantes (imagens com "ruído"). O sistema de Aleksander possui uma rede de 512 x 512 para a imagem e acima de 32.000 óctuplos, o que requer cerca de 8 milhões de bits (1 Mbyte) de RAM. Só há pouco tempo tais capacidades de memória se tornaram economicamente viáveis.

0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	1	0	1	1	0	0
0	1	1	0	0	0	1	0
0	0	1	1	1	1	0	0
0	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	0	0	0	1	0

0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0
0	0	1	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	1	1	1	0	0	0
0	1	1	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	0	0	0	0	1	0

A diferença entre R e A

Acima estão formas digitalizadas das letras A e R, mas nenhum dos padrões é bem definido: ambos apresentam manchas e distorções. Dados com esse tipo de "ruído" são um problema constante em todos os sistemas de inteligência artificial. Será que você conseguiria identificar as letras?

Processamento de figura

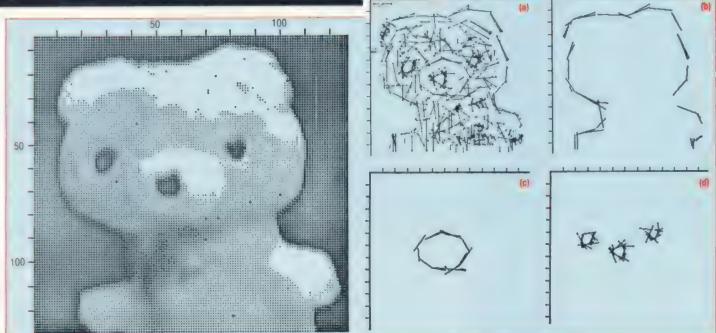
Esta foto de Vênus, obtida por uma sonda espacial, dá um exemplo de processamento de figuras. O computador acentuou os contornos entre áreas de diferentes altitudes. Tudo isso implica o uso de "cores falsas".

Na verdade, jamais poderíamos ver essa imagem sem a ajuda do computador, pois Vênus está sempre envolto em nuvens espessas, e nossos olhos não são sensíveis aos comprimentos de onda necessários para atravessá-las. Tais figuras mostraram-se muito úteis (às vezes, muito bonitas); baseiam-se nos comprimentos de onda do rádio, infravermelho e ultravioleta, e também da luz visível.

O leito oceânico também é mapeado pelo sonar, com o qual se obtêm "imagens acústicas", que se baseiam na sondagem por ultra-som, sem qualquer relação com as radiações eletromagnéticas (luz visível, infravermelho, ultravioleta etc.).

Outras modalidades de processamento de figuras são utilizadas para "limpar" imagens borradas e distorcidas. O uso desses algoritmos já se tornou rotineiro.

Esboços primários

Existe um método de processamento visual que não conta com o conhecimento prévio do que o padrão pretende representar. O esboço primário (a) extrai os contornos e a forma primitiva da imagem do ursinho de brinquedo por comparação das regiões adjacentes. Extratos adicionais do esboço primário — de (b) até (d) — exibem grupos importantes de elementos, que auxiliam no reconhecimento da imagem original.

BABEL INFORMÁTICA

As diversas linguagens de programação — desde as desenvolvidas na década de 50 e ainda em uso até as mais recentes — permitem ampla gama de opções para o processamento de quase toda espécie de dados.

Já foi demonstrado matematicamente que qualquer linguagem de computador pode simular o modo de operação de outras linguagens. Portanto, em termos dos problemas que podem resolver, todas elas se equivalem. Mas algumas se mostram muito mais adequadas para a resolução de determinado tipo de problema, enquanto outras possibilitam o desenvolvimento de programas mais eficientes.

Das inúmeras linguagens existentes a maior parte se volta para finalidades bem específicas — como o C, para a programação de sistemas, e o PROLOG, para a programação lógica. As chamadas linguagens genéricas não exigem do programador o controle dos recursos do equipamento (vale dizer: são "processuais", especificando apenas as operações que devem ser efetuadas pela máquina); constituem exemplos o FORTRAN, o COBOL, o APL etc.

Excluímos desta breve descrição as linguagens já tratadas com maior detalhe no decorrer da série, como o BASIC, o PASCAL e o ASSEMBLY.

Todas as linguagens consideradas podem rodar em qualquer equipamento, desde que este possua um programa tradutor apropriado, além de recursos — memória, por exemplo — suficientes. Deve-se levar em conta ainda que implementações para micros de várias dessas linguagens somente são encontradas no mercado internacional.

FORTRAN

Primeira linguagem de alto nível a ser amplamente utilizada, o FORTRAN (Formula Translator, "tradutor de fórmulas") foi desenvolvido pela IBM, na metade da década de 50. Trinta anos depois, continua sendo muito popular, especialmente entre engenheiros e cientistas.

Nesse período, apesar de sofrer duas padronizações (em 1962 e 1977), incorporando novas características e recursos, manteve a forma e o estilo originais.

Isso se deve à sua extraordinária capacidade de processamento numérico, constituindo uma das poucas linguagens capazes de manipular diretamente números complexos. Quase sempre compilado, o FORTRAN possui símbolos e regras sintáticas bastante semelhantes às convenções da matemática e das linguagens naturais (o inglês e o português, por exemplo). Os vários programas existentes em FORTRAN podem ser usados e adaptados para qualquer finalidade científica ou técnica; daí ser improvável que ele venha a se tornar obsoleto.

A linguagem dispõe de recursos como loops, ramificações, sub-rotinas, funções e atribuição de valores a nomes variáveis. Permite um número limitado de tipos de dados: números inteiros, números reais de precisão simples e dupla, números complexos, expressões booleanas e strings. Por outro lado, a única estrutura disponível para a organização dos dados é a tabela. As variáveis aceitam nomes de até seis caracteres, começando com uma letra. A menos que seja especificado de outro modo, todos os nomes de variáveis que comecem com as letras de I a N representam números inteiros.

No FORTRAN, um programa, uma sub-rotina ou uma definição de função sempre incluem uma instrução de cabeçalho (para identificação), uma série de instruções de especificação (para definição dos tipos de dados e declaração das dimensões da tabela), e, por fim, uma lista de comandos de execução (que constituem o corpo principal do programa).

COBOL

Existem mais linhas de programação escritas em COBOL (Common Business Oriented Language, "linguagem comum para atividades comerciais") do que em todas as outras linguagens juntas. Isso não se deve à sua prolixidade — mesmo programas simples exigem muitas linhas em COBOL —, mas por ser, desde sua criação, a principal linguagem para aplicações comerciais.

Rigidamente padronizado, o COBOL tem seu uso e desenvolvimento supervisionado por uma comissão permanente, a CODASYL (Conference On Data Systems Languages, "conferência sobre linguagens e sistemas de dados"). A própria linguagem foi projetada, em 1959, por essa comissão, que reuniu representantes do governo e dos principais fabricantes de computadores dos Estados Unidos.

Criado especialmente para aplicações comerciais e de gerenciamento, o COBOL caracteriza-se por programas compostos de palavras e frases, em linguagem natural, que podem ser decodificadas com facilidade por usuários sem qual-

quer formação técnica. A grande força da linguagem reside nos poderosos recursos para manipulação de arquivos, responsável pelo seu absoluto domínio na área empresarial.

O desenvolvimento e a operação de um programa em COBOL mostram diferenças em relação às outras linguagens, embora possamos encontrar muitos dos conceitos existentes em todas as linguagens processuais. Os programas apresentam-se em quatro divisões, conhecidas pelos nomes identification, environment, data e procedure division.

A identification division inclui os nomes do programa e do programador, as datas em que foi escrito e compilado, bem como os comentários introdutórios. A environment division foi planejada inicialmente para ser dependente da máquina, especificando detalhes do computador no qual o programa foi escrito e executado. Muitos desses detalhes encontram-se hoje sob controle do sistema operacional, e o principal uso dessa divisão é a especificação dos arquivos externos usados no programa. A data division contém detalhes de todas as áreas de dados usadas pelo programa. Ela compreende duas seções principais: uma que abriga a disposição dos registros para os arquivos externos, e outra para os dados usados apenas num programa.

As operações e procedimentos a executar com os dados são especificados na procedure division, que, se necessário, pode ser dividida em seções. Cada uma destas, ou toda a divisão, compõe-se de parágrafos formados por sentenças. Uma sentença pode corresponder a um ou mais comandos. A estrutura da procedure division foi projetada para ficar tão próxima quanto possível do inglês. Isso, juntamente com os identificadores que chegam a ter trinta caracteres, torna os programas compreensíveis até para leigos em programação.

ALGOL

Resultado de um programa internacional para a elaboração de uma linguagem algorítmica padronizada, o ALGOL (Algorithmic Language, "linguagem algorítmica") teve sua primeira versão apresentada em 1958. Contudo, ainda sofreu várias reformulações e aperfeiçoamentos até o lançamento da especificação oficial, em 1960.

Trata-se de uma linguagem de aplicação genérica. Bastante poderosa, permite a expressão precisa de procedimentos numéricos. Caracteriza-se por utilizar uma notação algébrica completa e tem como finalidade a solução de problemas lógicos por meio de algoritmos elegantes e eficientes. Embora semelhante ao FORTRAN, o ALGOL dispõe de maiores recursos e possui uma estrutura extremamente formalizada, que possibilita a eliminação de exceções e maior legibilidade.

Apesar de não ser tão famoso quanto o FORTRAN e o COBOL, ocupa uma posição destacada na evolução das linguagens de programação. Foi a partir de uma versão do ALGOL que Niklaus Wirth desenvolveu, em 1968, a linguagem PASCAL.

Um dos motivos que impediram a popularização do ALGOL é o amplo uso que ele faz de símbolos pouco comuns, além da complexidade cada vez maior que foi adquirindo de uma versão para outra. Suas implementações restringem-se a pesquisas universitárias, especialmente na Europa.

LISP

Desenvolvida no período de 1956 a 1958, a linguagem LISP (List Processor, "processador de listas") não apenas continua em uso, como vem se tornando cada vez mais importante nas pesquisas de inteligência artificial. Isso em função de sua impressionante capacidade para manipular listas.

O LISP foi projetado especificamente para processar dados não numéricos cujo comprimento e estrutura, muitas vezes, variam durante o processamento. Nessa linguagem, define-se uma lista como um conjunto de itens em determinada ordem entre parênteses. Por isso, o LISP oferece um modo conveniente para a representação de dados, como frases em linguagens naturais, fórmulas matemáticas, teoremas lógicos ou até mesmo programas de computadores. Esse é um dos conceitos mais poderosos do LISP, e tanto os dados como os programas são representados do mesmo modo. Isso facilita muito a criação de programas que desenvolvem e rodam outros programas.

Um programa em LISP nada mais é do que uma coleção de funções, escrita em forma de lista. Quase tudo nessa linguagem se faz por meio de funções, o que a torna especialmente apta para aplicações, tais como: geração e verificação de provas matemáticas, reconhecimento de padrões, processamento algébrico, simulação de formas humanas de resolução de problemas, programação heurística, análise lingüística e exploração e desenvolvimento de novas linguagens de programação não processuais.

Outra característica poderosa do LISP é a sua extensibilidade. Significa que ele permite ao programador acrescentar recursos que se tornam parte integrante da linguagem.

PL/1

O PL/1 (Programming Language 1, "linguagem de programação 1") foi lançado em 1965 como uma linguagem de programação genérica, voltada para a resolução de problemas tanto comerciais como científicos. Combinando recursos do FORTRAN (por exemplo, instruções simples e concisas) e a capacidade de manipulação de arquivos do COBOL, o PL/1 possui uma eficiência somente comparável à sua complexidade.

A linguagem se baseia no uso de estruturas chamadas procedimentos — blocos de instrução que executam as funções desejadas. Um determinado procedimento pode ser incorporado hierarquicamente a outro, e quaisquer dados declarados ficam automaticamente disponíveis para todos os procedimentos subsidiários. O PL/1 dispõe de recursos para a manipulação de strings alfanuméricos ou strings de bits. Além disso, permite que os programadores descrevam os dados em termos de tabelas e outras estruturas "apontadoras".

Apesar de sua impressionante versatilidade, a linguagem PL/1 não encontrou muita aceitação. A complexidade foi um dos empecilhos à sua difusão: a maioria das implementações emprega mais de duzentas palavras-chaves. Além disso, apenas uma pequena parte de seus recursos é utilizada na maioria das aplicações, o que certamente desestimulou os usuários que já conheciam o FORTRAN e o COBOL.

APL

Elaborado por Kenneth Iverson, em 1967, a partir de uma notação especial para o processamento de equações diferenciais aplicadas a modelos econômicos, o APL (A Programming Language, "uma linguagem de programação") logo se tornou motivo de muita polêmica.

Muitos o consideram a linguagem processual mais flexível, poderosa e concisa que existe. Mas seus adversários criticam-lhe a sintaxe e a pouca adequação ao processamento de tarefas "produtivas" e de programas que envolvam muita edição.

Utilizando uma notação muito semelhante à da matemática e operando em modo interativo, o usuário do APL com algum treino em ciências exatas não encontra dificuldade para obter resultados com seus programas iniciais.

Dentre as linguagens processuais de aplicação genérica, o APL distingue-se por possuir o mais amplo conjunto de operadores primitivos. Por meio deles, o programador pode executar funções como: geração de números aleatórios, cálculo fatorial, formação e inversão de matrizes etc. Esses operadores, acionados por uma única tecla, realizam tarefas que em outras linguagens exigiriam dezenas de comandos.

O APL é muito popular entre matemáticos e estatísticos devido à velocidade com que os algoritmos podem ser desenvolvidos e testados. No entanto, ele não se restringe às aplicações numéricas, sendo também usado numa ampla gama de aplicações comerciais, tais como produção de documentos, análises de gráficos e análises financeiras. Além disso, mostra-se particularmente apropriado para trabalhos de pesquisa e aplicações educacionais.

PROLOG

O PROLOG (Programming Logic, "lógica de programação") começou a ficar conhecido depois que os japoneses o escolheram para o desenvolvimento dos computadores inteligentes de quinta geração. Constitui, ao lado do LISP, uma das ferramentas básicas para as pesquisas de inteligência artificial.

Linguagem recente, o PROLOG foi inventado no início da década de 70 pelo francês Alain Colmerauer, da Universidade de Marselha. Baseia-se na lógica dos predicados, semelhante à que usamos na vida cotidiana para a resolução de problemas, embora possua notação específica e determinadas regras. Define-se um predicado como uma relação entre coisas. Na frase "João ama Maria", o predicado é "ama". Em PROLOG, isso seria escrito da seguinte maneira: "ama (João, Maria)". Embora a leitura seja mais difícil, essa notação evidencia o predicado e seus argumentos.

Por meio da lógica dos predicados, pode-se descrever o mundo em termos de fatos e implicações. E, a partir desses fatos, deduzir outros. No PROLOG, isso é feito por intermédio das variáveis lógicas, que se assemelham àquelas existentes no BASIC e em algumas outras linguagens de programação.

Foi visando proporcionar ao programador um modo de descrever a estrutura lógica de um programa por meio de um banco de dados (com fa-

tos) e regras lógicas que se projetou o PROLOG. Sua sintaxe é bastante simples, mas a terminologia muito confusa. O elemento mais importante — o termo — pode ser uma constante, uma variável ou uma estrutura. As estruturas são criadas a partir de constantes e variáveis. Uma estrutura muito comum nos programas em PROLOG, conhecida como fato, compõe-se de um predicado isolado ou seguido por uma lista de argumentos.

O PROLOG, embora seja considerado uma linguagem não processual (o programador só precisa descrever o problema, pois a linguagem faz o resto), possui mecanismos rígidos para resolução de problemas, os quais não são facilmente modificados.

De qualquer modo, trata-se de um código particularmente adequado ao processamento de linguagens naturais. E, como essa área é um aspecto essencial das verdadeiras máquinas user-friendly ("amigável ao usuário"), fundamental para as pesquisas de computadores de quinta geração, o PROLOG tem seu lugar assegurado no desenvolvimento tecnológico do final do século XX.

C

Linguagem de programação genérica, o C não foi projetado para resolver um tipo específico de problema. Não se trata, portanto, de uma linguagem de alto nível, como o BASIC, o PASCAL e outras; nem de uma linguagem de baixo nível, como o ASSEMBLY.

Os programadores de sistemas são os principais usuários do C, que reúne as vantagens das linguagens de alto e de baixo nível. Além disso, ele não se limita a um sistema operacional determinado, possibilitando a utilização de seus programas em computadores que usam os mais diversos processadores.

Em virtude da feroz competição para lançar novos produtos no mercado, essa versatilidade se mostra fundamental para as indústrias de microcomputadores, em especial quanto ao desenvolvimento de software para os novos modelos de 16 bits. Afinal, um programa recém-desenvolvido precisa ser rapidamente adaptado aos quatro ou cinco principais microprocessadores. Com a linguagem C, apenas aqueles trechos do programa que usam código de máquina específico precisam ser reescritos, mantendo-se inalterada a maior parte do programa. Por isso, o C vem atraindo cada vez mais atenção dos principais fabricantes de software.

Tal como algumas linguagens de alto nível, o C foi projetado no sentido de permitir uma programação estruturada. Empregando blocos e nomes flexíveis para programas e variáveis, facilita a elaboração, leitura e depuração. Ao mesmo tempo, o usuário tem à sua disposição dezenas de operadores de baixo nível, que correspondem exatamente às operações em linguagem de máquina de vários microprocessadores. Com o uso de tais comandos, uma linha em código C pode substituir várias linhas em código de máquina.

Outra vantagem do C está no fato de seus programas serem executados com muito mais rapidez do que a possibilitada por qualquer linguagem de alto nível; além de ser mais facilmente aprendida do que o ASSEMBLY.

Código extremamente compacto, atua com cerca de trinta palavras reservadas. Todas as versões são escritas na própria linguagem (e não em ASSEMBLY), principal motivo de sua transportabilidade e facilidade de implementação em outros sistemas.

Com regras bastante flexíveis, em geral C permite ao programador fazer o que quiser, desde que não haja proibição explícita. A linguagem foi desenvolvida de forma a pressupor um programador profissional. Por exemplo, ela permite a manipulação de bits individuais; os apontadores têm acesso direto às posições de memória, sendo possível incluir ou definir novas estruturas de dados quando as disponíveis revelam-se insuficientes. Por outro lado, embora tais recursos impliquem considerável aumento de potência e flexibilidade dos programas escritos em C, eles também aumentam a possibilidade de erros obscuros difíceis de localizar e corrigir.

Desde que foi criado, em 1972, por Dennis Ritchie, da Bell Laboratories, o C tem sido associado ao sistema operacional Unix. Desenvolvidos originalmente para o minicomputador PDP 11, da Digital Equipment Corporation, tanto o C quanto o Unix foram implementados em todos os tipos de computadores, dos micros ao supercomputador Cray-1.

Por haver sido criado para avançados sistemas que operavam em modo compartilhado, o Unix revelou-se flexível e poderoso o suficiente para atender às necessidades dos programadores profissionais que trabalham no desenvolvimento de novas gerações de microcomputadores, como os de 16 e 32 bits. O Unix foi escrito em C, o que o torna transportável para uma grande variedade de microprocessadores.

MEDIDATA

Mesmo produzindo mínis, micros e supermicrocomputadores, a Medidata adota como filosofia de trabalho o assessoramento aos clientes. Sua preocupação é oferecer soluções "sob medida".

Criada em outubro de 1976, a Medidata no princípio atuava apenas como bureau de prestação de serviços. Isso era feito através de terminais interligando seus clientes à central de serviços, que já naquela época utilizava minicomputadores.

No ano seguinte ao de sua fundação, quando o governo brasileiro passou a enfocar a política nacional de informática de maneira mais dedicada e consistente, a companhia tratou logo de garantir uma fatia nesse mercado. Para tanto, uniu sua especialização em prestação de serviços ao desenvolvimento de tecnologia própria de software, para otimizar o desempenho de seus futuros equipamentos. Em 1978, assim que a empresa se viu reestruturada de forma segura, lançou seu primeiro computador, o sistema M2001.

Mesmo tendo ramificado as atividades, a filosofia de trabalho da Medidata continua sendo o atendimento direto aos clientes. Seu sistema de trabalho é conhecido na empresa como "lógica sob medida". Refere-se ao tratamento especializado no tocante à automação dos departamentos de cada empresa-cliente, em vista das diferentes necessidades de cada uma em função do tamanho, do ramo de atividade e da filosofia.

O M301
O microcomputador profissional da Medidata. Embora com 16 bits, opera também 8, ampliando sua faixa de utilização.

Seguindo à risca sua proposta de trabalho, a Medidata desenvolve projetos e softwares básicos, de apoio e aplicativos, fabrica computadores, presta serviços de consultoria e desenvolvimento de projetos especiais, além da prestação de serviços de apoio, manutenção e treinamento.

Para atender seus clientes, a companhia possui uma estrutura própria compatível com as necessidades de manutenção, assistência técnica e de suporte aos equipamentos por eles adquiridos. A empresa dispõe de engenheiros, técnicos e analistas distribuídos por uma rede de filiais e centros de atendimento. Para agilizar esse serviço foi criado o Telessuporte, onde o cliente recebe um diagnóstico e as medidas corretivas para seu equipamento através de linhas telefônicas e terminais remotos.

Os centros de treinamento estão equipados com recursos audiovisuais, salas de aula, professores e aparelhagem técnica. Neles os clientes da Medidata recebem instruções visando maior familiarização com os diversos produtos comercializados pela empresa. Usuários com um tipo de problema específico são atendidos por uma equipe voltada para o projeto e implementação de sistemas aplicativos. Esse grupo desenvolve soluções especiais, adequadas a cada necessidade. Entre os clientes já atendidos por essa modalidade de serviços incluem-se a Rede Globo de Televisão, a Esso Brasileira de Petróleo e o Bank of Boston.

Para atender a um mercado crescente, com uma demanda vertiginosa de novos produtos e serviços, a Medidata oferece uma série de programas aplicativos, básicos e de apoio. Esses pacotes foram desenvolvidos de forma integrada e modular, e cobrem as funções básicas de administração financeira, de materiais e de recursos humanos. Como as necessidades variam de acordo com a filosofia de trabalho de cada empresa, os aplicativos da Medidata podem ser modificados e ajustados pelo próprio usuário ou pela Medidata.

O Gerenciador de Aplicações, um programa de apoio desenvolvido pela empresa, permite acelerar o desenvolvimento de aplicações e aumentar a produtividade. O software básico é o MUMPS, que pode ser utilizado em todos os seus computadores, favorecendo a compatibilidade entre os vários aparelhos.

Na linha dos equipamentos de uso profissional, a Medidata fabrica mínis, micros e supermicros. O M301 é um microcomputador de 16 bits, compatível com o IBM PC e com o micro CP/M80 de 8 bits, que utiliza o sistema MUMPS. O multiusuário M1001, um supermicrocomputador, possibilita a interligação de até dez terminais, além de compartilhar impressoras e discos, e permitir comunicação com a rede nacional de telex. No terreno dos mínis, são produzidos o M2001, interligando até vinte terminais, locais ou remotos, com 384 Mbytes de capacidade em discos, e também o M3001, para até 64 terminais, locais ou remotos, e 768 Mbytes de memória em disco.

APRENDENDO A APRENDER

Os pesquisadores da inteligência artificial procuram desenvolver "sistemas-criança", que aprendem por si mesmos. Um exemplo é o BEAGLE, um procriador de regras baseado na teoria da seleção natural.

Se um sistema de computador melhorou seu desempenho numa tarefa, durante um certo tempo, sem ser reprogramado, é justo dizer que aprendeu. Isso implica a presença de um padrão para o sistema. Sem uma medida comparativa, não se pode falar em aprendizado.

Um algoritmo de aprendizagem procura realizar um ou mais dos seguintes objetivos:

- Cobrir uma faixa maior de problemas;
- Dar soluções mais precisas;
- Obter respostas com um mínimo de dispêndio; e
- Simplificar o conhecimento codificado.

Os sistemas de aprendizagem mais bem-sucedidos têm sido aplicados a problemas de classificação, nos quais se devem examinar os dados de entrada e, de algum modo, classificá-los e identificá-los ou interpretá-los. Além disso, os métodos em que estamos interessados devem ser capazes de generalizar, oferecendo uma resposta apropriada a uma nova situação.

Qualquer sistema de aprendizado de máquina deve incluir os seguintes componentes:

- Um conjunto de estruturas de dados, que representa o nível de habilidade do sistema num determinado momento (as regras);

Intelecto crítico

A maneira pela qual uma criança aprende é complexa e só parcialmente compreendida. Alguns psicólogos sugerem que elas aprendem compondo "esquemas" — estruturas de regras —, muitas vezes por um processo de tentativa e erro: testando hipóteses (as regras) e conservando as que dão resultados corretos. Em muitas situações, o processo de aprendizagem é orientado pelo crítico (o professor), que ajuda a criança (o aprendiz) a avaliar e refinar o conjunto interno de regras que irão ajudá-la a melhorar seu desempenho numa tarefa. Os sistemas de aprendizagem tentam reproduzir essa assistência, criando uma base de conhecimento que pode ser utilizada juntamente com exemplos de treinamento e um método de avaliar o desempenho de cada regra.

Sistema Ideal
Saída Correta
Desejada
Real
Entrada
Executante
Saída
Regras (Base de conhecimento)
Crítico
Aprendiz

UM MECANISMO DE APRENDIZADO

IDEAL
ENTRADA
SAÍDA
EXECUTANTE/ BASE DE REGRA/ APRENDIZ
CRÍTICO

quadrado
ENTRADA
?
(paralelogramo)
Aluno
tem 4 lados
Professor
FASE 1

quadrado
?
(losango)
Aluno
1) um quadrado tem 4 lados
Professor
são necessárias mais regras
FASE 2

quadrado
?
(quadrado sobre uma ponta)
Não é quadrado
Aluno
1) um quadrado tem 4 lados
2) todos os lados têm o mesmo tamanho
3) dois lados são verticais
Professor
FASE 3

quadrado
?
SAÍDA CORRETA
Aluno
Professor
correto
FASE 4

- Um algoritmo de tarefa (o executante), que utiliza as regras para guiar a atividade de resolução de problemas;
- Um módulo de retroalimentação (o crítico), que compara resultados obtidos e metas propostas;
- O próprio mecanismo de aprendizagem (o aprendiz), que utiliza a retroalimentação do crítico para aperfeiçoar as regras.

Mas como aperfeiçoá-las? O BEAGLE (Biological Evolutionary Algorithm Generating Logical Expressions, "algoritmo biológico evolutivo gerando expressões lógicas") é um sistema para computador que produz regras de decisão por indução, a partir de um banco de dados. *Beagle* era o nome do navio em que Charles Darwin realizou a viagem de circunavegação do globo terrestre que lhe deu subsídios para a teoria sobre a evolução das espécies — e o sistema baseia-se no princípio darwiniano da seleção natural. As regras que não concordam com os dados vão sendo eliminadas e substituídas por "mutações" melhores ou por outras criadas pelo "acasalamento" com regras mais bem adaptadas. As regras são expressões booleanas representadas por estruturas em árvore.

O software original consistia em dois programas em PASCAL, denominados HERB (Heuristic Evolutionary Rule Breeder, "reprodutor de regras evolutivas heurísticas") e LEAF (Logical Evaluator Aid Forecaster, "avaliador e previsor lógico"). O HERB requer três arquivos de entrada, a serem criados pelo usuário: um de dados, outro de pagamentos e um terceiro para as regras velhas. O primeiro contém o conjunto de treinamento, para o qual se conhece a correta categoria de pertinência como membro. O usuário também precisa fornecer uma matriz de pagamentos, que define o valor ou custo das classificações corretas e incorretas.

O LEAF é mais simples: apenas apanha um arquivo de dados do mesmo formato que o conjunto de treinamento e roda sobre ele um arquivo de regras. Pode ser incumbido de gravar, entre outras coisas, uma lista ordenada de itens do arquivo de dados, começando com os de maior probabilidade de satisfazer a uma dada classe (tal como é definida pelo arquivo de regras).

O algoritmo de aprendizagem BEAGLE consiste em repetir para várias "gerações" (uma geração corresponde a uma rodada completa através dos dados de treinamento) o seguinte procedimento:

1. Avaliar cada regra em cada amostra, de acordo com a matriz de pagamentos, com um bônus concedido às regras mais curtas.
2. Pôr as regras em ordem decrescente de mérito e remover a metade inferior.
3. Substituir regras "mortas", acasalando um par de sobreviventes escolhidos aleatoriamente, de modo a recombinar porções de regras boas.
4. Produzir mutações em regras escolhidas aleatoriamente (mas não na que está no topo do arquivo) e aplicar um procedimento TIDY às novas regras — prontas para a geração seguinte.

O procedimento TIDY elimina duplas negações e outras redundâncias sintáticas eventualmente geradas, "podando" a árvore de regras.

O resultado é que apenas as regras mais bem adaptadas sobrevivem para produzir a geração seguinte, e o arquivo de regras "evolui" por sucessivas gerações — rodadas de treinamento.

Charles Darwin (1809-1882)
As teorias de Darwin sobre a evolução sustentam que as espécies se adaptam e se aperfeiçoam, em resposta aos ambientes em que vivem, através da seleção natural; apenas os indivíduos mais fortes e mais bem adaptados sobrevivem, para dar origem à geração seguinte e passar adiante suas características. Essa idéia vem sendo aplicada com sucesso à aprendizagem das máquinas, melhorando o desempenho da inteligência artificial nos problemas de ordenamento; desenvolve, para isso, conjuntos de regras de classificação. Não é por acaso que um desses sistemas chama-se BEAGLE, nome do navio em que o então jovem naturalista Charles Darwin deu a volta ao mundo, recolhendo subsídios para sua futura teoria sobre a origem das espécies.

Aprendizagem evolutiva

Vejamos, agora, o que acontece quando colocamos em prática uma abordagem darwiniana da aprendizagem de máquina.

Vamos examinar uma versão do famoso PCV (Problema do Caixeiro Viajante). Ele possui uma tabela com a distância entre as 48 capitais de Estados da área continental dos Estados Unidos (excluídos o Alasca e o Havaí); deve visitar cada uma e voltar ao ponto de partida. O objetivo consiste em minimizar a distância total percorrida.

Parece desconcertante de tão simples, mas o número de percursos é de N – 1, sendo N o número de cidades. Para 48 cidades, existem 47 primeiros passos possíveis, seguidos por 46 segundos passos possíveis etc. Na verdade, há mais rotas potenciais do que átomos no universo!

O PCV pode ser tratado como um problema de busca. Também se pode manipulá-lo pelo chamado método de Monte Carlo, que utiliza tentativa e erro aleatoriamente. Entretanto, vamos abordá-lo como um problema de aprendizagem. Não esperamos resgatar a solução ótima, porém uma boa solução.

O programa GENE (General Evolutionary Network Explorer, "explorador geral de rede em evolução"), aqui apresentado, é um sistema de aprendizagem especialmente adaptado para explorar redes e desenvolver regras que evoluem de modo a produzir rotas cada vez mais econômicas através da malha. Antes de descrevê-lo, precisamos desenvolver uma rede para ele explorar. Um bom exemplo pode ser extraído de O Guia da Galáxia para o Penetra, que lista todos os planetas, num raio de 80 anos-luz, onde você pode encontrar uma festinha decente numa noite de sábado. O objetivo do jogador é comparecer a vinte festas e retornar ao ponto de partida na mesma noite. Para isso, recorre a um mapa que apresenta as vias principais no hiperespaço galáctico, indicando o tempo que se gasta para viajar entre duas paradas quaisquer. Aos pares de nodos que não estejam conectados pelo hiperespaço atribui-se arbitrariamente um dispêndio de travessia igual a 1.000 unidades de tempo.

Num esquema clássico de aprendizagem evolutiva, trata-se uma população de estruturas-regras como "pseudo-organismos". Cada regra define uma solução potencial para o problema. São também usadas para gerar novas estruturas-regras sob formas que imitam algumas características da reprodução biológica — por exemplo, a herança genética.

Uma boa estirpe
O GENE cria novos caminhos na rede acasalando duas rotas bem-sucedidas. Como estas correspondem a strings de vinte caracteres, escritos em BASIC, o processo de produzir descendência é essencialmente uma simples manipulação de strings. Neste exemplo, o substring HIBCD é extraído do pai doador e emendado ao pai receptor, assegurando-se de que nenhuma letra seja duplicada. O processo garante, à semelhança do que ocorre nos cruzamentos genéticos reais, a transmissão das características dos pais aos seus descendentes.

Dependendo de seu desempenho, selecionam-se as regras suscetíveis de sobreviver por mais tempo e com maior probabilidade de procriação. No GENE, as regras conservam a forma de strings no programa em BASIC. Na cadeia

ABJHMNCDKTSFRQEGILOP

cada letra denota um planeta na rede e ocorre uma só vez no string. Isto é, um string de vinte caracteres define um percurso particular pela rede. Somando as distâncias para se chegar aos planetas na ordem indicada, podemos avaliar cada string. Quanto menor a combinação das distâncias, melhor o percurso.

Convencionamos que os strings piores que a média sejam apagados após cada "geração". Como substituí-los? O mecanismo de "reprodução sexual" não é o único meio de gerar novos strings: cerca de 8% dos sobreviventes são "mutantes".

A sub-rotina das mutações, que começa na linha 3.500, apenas executa um certo número de permutações aleatórias. A mutação é um operador genético que age em segundo plano, impedindo que o sistema fique preso a uma solução boa, mas passível de melhoria. O padrão médio da população de regras realmente melhora com o tempo, embora isso não seja uma progressão contínua (até as melhores regras podem "morrer").

Nesta altura, colocam-se duas questões interligadas. Até que ponto o método é bom? Por que ele funciona? Para respondê-las, devemos compará-lo a uma típica abordagem Monte Carlo — os algoritmos genéticos são métodos de Monte Carlo modificados. Com esse procedimento, são geradas soluções aleatórias e conservada a melhor solução, dentro de um prazo especificado. Neste exemplo, seria necessário testar uma regra-string aleatória, avaliá-la, conservá-la se for a melhor e fazê-la sofrer mutação. O processo pode ser repetido à vontade.

Engenharia alfabética

Em primeiro lugar, duas estruturas parentais são escolhidas aleatoriamente entre as que sobreviveram ao processo de seleção. Uma delas é denominada "doador" (R1%) e a outra (R2%), "receptor" (ver as linhas 3.030 e 3.050). Extrai-se aleatoriamente uma lasca de "material genético" (um substring) do doador e coloca-se o string S$. Chama-se então a sub-rotina 3.300, que emenda a lasca do doador no receptor, deslocando todos os caracteres do string receptor (exceto os que provieram do doador) na ordem em que aparecem. O processo de acasalamento é, portanto, assimétrico. A união de X com Y não dá o mesmo resultado que a de Y com X, mesmo que as posições aleatórias P1% e P2% sejam idênticas. Vamos examinar um exemplo em escala reduzida (o string "real" tem vinte caracteres). Nos pais

Doador A G H I B C D F E J
Receptor H G A B F J I C D E

podemos selecionar o substring

H I B C D

como sendo a contribuição do doador e uni-lo às letras restantes do receptor, para produzir

H I B C D G A F J E

Deve-se salientar que esse não é o único modo de cruzar um par de regras-strings. Você pode imaginar outros, mas deve garantir — tal como ocorre nos cruzamentos genéticos reais — que segmentos de informação parental sejam transferidos à geração seguinte. Também deve garantir que cada acasalamento produza uma descendência "válida". Significa dizer que a rotina de limpeza na linha 4.000 do programa GENE nada tem para fazer; foi incluída unicamente para que o programa ficasse completo.

Esse tipo de programa descobrirá rapidamente uma boa solução — mas a melhor será mais lenta. Após 20.000 tentativas, é provável que a solução seja apenas um pouco melhor — se o for — do que a obtida após 10.000 tentativas.

O método de Monte Carlo é, assim, uma busca cega. Um algoritmo genético, por sua vez, mobiliza tudo o que encontra para ser orientado depois em sua busca. Padrões que contribuem para um bom desempenho são preservados e propagados por toda a população de regras. Disso resulta, na maioria dos casos, uma razoável estratégia de busca.

O programa GENE

```
REM ************
REM ** GENE **
REM ************
GOSUB 1000: REM MONTA A MATRIZ
[illegible]
INPUT "QUANTAS GERACOES [illegible]
GOSUB 1700: REM REGRAS INICIAIS
IF SN = 0 THEN INPUT "NO INICIAL E O NUMERO ";SN
REM ***LOOP PRINCIPAL***
[illegible]
PRINT "GERACAO ";G
GOSUB [illegible]: REM AVALIA REGRAS
GOSUB [illegible]: REM ABANDONA AS REGRAS INADEQUADAS
[illegible] REM AGRUPA AS BOAS REGRAS
GOSUB 3500: REM MUTACAO
GOSUB 4000: REM ORDENA
IF G < MG THEN 120
GOSUB 5000: REM INTRODUZ NOVAS REGRAS
END

REM ROTINA PARA MONTAR O MAPA
SI = 20:NR = 24
DIM N$(20),L(20,20)
DIM N(SI)
DIM R$(NR),RV(NR)
REM REGRAS E VALOR DAS REGRAS
FOR I = 1 TO SI
FOR J = 1 TO SI
L(I,J) = 1000: REM DEFEITO
IF I = J THEN L(I,J) = 0
NEXT J: NEXT I
NC = DTL = 0
FOR I = 1 TO NR:RV(I) = 0: NEXT I
RESTORE
REM LER NOME E NUMERO DO NO
READ N$,ID
PRINT N$,ID
IF ID < > 0 THEN GOSUB 1500
IF ID < > 0 THEN 1040
PRINT NC;" LOCALIZACOES LIDAS"
PRINT L;" MALHAS SEM DEFEITO"
RETURN

1500 REM NO INDIVIDUAL E CONEXOES
1510 NC = NC + 1
1520 IF ID < > NC THEN PRINT "CUIDADO NO ";ID;" NAO FUNCIONA"
1530 N$(NC) = N$
1550 REM ***MONTA A MATRIZ DAS DISTANCIAS NODAIS***
1560 READ NI,NT
1570 L(ID,NI) = NT
     REM MALHAS ZERO NAO INTERESSAM
1590
1600 IF NI > 0 THEN 1550
1610 RETURN
1620
1700 REM REGRAS INICIAIS SIMULADAS
1710 S$ = LEFT$ ("ABCDEFGHIJKLMNOPQRSTUVWXYZ",SI)
1720 FOR R = 1 TO NR
1730 R$(R) = S$
1740 PRINT S$,R
1750 I = INT ( RND (1) * SI + 1):J = INT ( RND (1) * SI + 1)
1755 IF I = 1 OR J = 1 GOTO 1750
1760 GOSUB 6000: REM TROCA
1770 NEXT R
1775 RETURN
1777
2000 REM ROTINA DE AVALIACAO DAS REGRAS
2010 T = 0
2020 FOR R = 1 TO NR
2030 IF RV(R) < = 0 THEN GOSUB 2200
2040 T = T + RV(R)
2050 NEXT R
2060 AV = T / NR: REM VALOR MEDIO
2070 PRINT "PONTUACAO MEDIA ";AV
     [illegible]
     :
     REM AVALIACAO DE REGRA UNICA
     S$ = R$(R)
     P1 = SN: REM NO INICIAL
     [illegible]
     FOR S = 1 TO LEN (S$)
     P2 = ASC ( MID$ (S$,S,1)) - 64
     IF P2 = SN THEN GOTO 2290
     DT = DT + L(P1,P2)
     P1 = P2
     NEXT S
     RV(R) = DT + L(P2,SN)
     RETURN

     REM ROTINA PARA ABANDONAR REGRAS [illegible]
     FOR R = 1 TO NR
     IF RV(R) < B THEN B = RV(R):B$ = R$(R)
     IF RV(R) > AV [illegible] ( INT ( RND (1) * 100 + 1)) > 88 THEN R$(R) = "[illegible]
     RV(R) = 0
     NEXT R
     RETURN
     REM VALORES MENORES SAO PREFERIVEIS

     REM ROTINA DE DIVISAO DE GENE
     FOR R = 1 TO NR
     IF RV(R) ) 0 THEN GOTO 1120
     R1 = INT ( RND (1) * NR + 1): IF RV(R1) < = 0 THEN [illegible]
     R2 = INT ( RND (1) * NR + 1): IF RV( [illegible] ) < = 0 THEN GOTO [illegible]
     REM [illegible]
     P2 = INT ( RND (1) * (SI [illegible] 1) + 1)
     P1 = INT ( RND (1) * SI + 1): REM DIVISAO DE [illegible]
     S$ = MID$ (R$(R1),P1,P2)
     GOSUB [illegible]: REM DIVIDE O RESTANTE
     R$(R) = S$
     NEXT R
     RETURN

     REM ROTINA DE DIVISORA DE GENE
     FOR N = 1 TO [illegible]
     N(S) = 0: [illegible]
     FOR [illegible] = 1 TO LEN (S$)
[illegible]
     [illegible] N(X) + 1
[illegible]
     FOR [illegible] = 1 TO LEN (R$(R1))
     X = ASC ( MID$ (R$(R2),S [illegible]
     IF N(X) > 0 THEN GOTO [illegible]
     S$ = S$ + MID$ (R$(R2),S, [illegible]
     [illegible] = N(X) + 1
     NEXT
     RETURN

     REM ROTINA DE MUTACAO
     FOR R = 1 TO NR
     IF ( RND (1) * 100) > B THEN GOTO 3580
     S$ = R$(R)
     [illegible]
     R1 = INT ( RND (1) * SI + 1): IF R1 = 1 THEN GOTO [illegible]
     [illegible] INT ( RND (1) * SI + 1): IF R2 = 1 THEN GOTO 3540
     [illegible] R2: GOSUB 6000: REM TROCAR
     IF I = 1 OR J = 1 THEN 3530
     NEXT I
[illegible]
     [illegible]
     REM [illegible] SOMENTE AGORA
     REM [illegible] NECESSITA DE [illegible]
[illegible]

     REM ROTINA ORDENADORA
     RETURN
     REM INTRODUZ AGORA
     :
     REM SAIDA DOS RESULTADOS
     PRINT "OS CAMINHOS SAO:"
     BB = SI * 1000
     B = 0
     FOR I = 1 TO NR
     IF RV(I) = 0 THEN 5100
     PRINT I,RV(I)
     PRINT R$(I)
     IF RV(I) < BB THEN BB = RV(I):B = I
     NEXT
     GET A$: IF A$ = " " THEN 5105
     PRINT
     [illegible] "O MELHOR E :"
     PRINT R$(B),B
     S$ = R$(B): GOSUB 6400
     [illegible] "DISTANCIA = ";RV(B)
     GET A$: IF A$ = " " THEN 5114
     PRINT "O MELHOR DE TODOS E:"
     [illegible]
     [illegible] = B$: GOSUB 6400
     PRINT "DISTANCIA = ";B
     GET A$: IF A$ = " " THEN 5144
     PRINT
     RETURN

     REM ***TROCA DOIS CARACTERES EM S$***
     IF I > J THEN T = I:I = J:J = T
     X$ = MID$ (S$,I,1)
     Y$ = MID$ (S$,J,1)
6070 S$ = LEFT$ (S$,(I - 1)) + [illegible]
6080 S$ [illegible] + MID$ (S$,J + 1)
     RETURN

6400 REM ***[illegible]***
6410 PRINT [illegible]
6440 FOR I = 1 TO LEN (S$)
     N = ASC ( MID$ (S$,I,1)) - 64
6460 IF N < > SN THEN PRINT I;" ";N$(N)
     NEXT
     [illegible]
6490 RETURN
6500 :
8000 REM DADOS PARA O MAPA PLANETARIO
8010 DATA HELIOSOL, 1
8020 DATA 2,4,17,30,20,60, 0,0
8030 DATA [illegible]
     DATA 1,4, 3,6, 0,0
8050 DATA AFRRODITE,3
     DATA 2,6, 5,7, 0,0
8070 DATA LUNA, 4
     DATA 5,1, 6,10, 0,0
     DATA TERRA FIRMA, 5
     DATA 3,7, 7,8, 6,8, 4,1, 0,0
8110 DATA ELMER KINGDOM, 6
     DATA 5,8, 7,1, 9,12, 8,12, 3,10, 0,0
     DATA EBORIA, 7
8140 DATA 5,8, 9,11, 8,11, 6,1, 0,0
     DATA EUREKA, 8
     DATA 6,12, 7,11, 9,1, 10,16, 11,25, 0,0
8170 DATA [illegible]
     DATA 10,15, 8,1, 6,12, 7,13, 0,0
8190 DATA TITANIUM CITY, 10
8200 DATA 7,15, 12,24, 11,20, 8,16, 0,0
8210 DATA [illegible], 11
     DATA 8,25, 10,20, 12,17, 11,20, 0,0
     DATA TRIDENT, 12
8240 DATA 10,24, 14,22, 13,20, 11,17, 10, [illegible] 0,0
     DATA LIMBO, 13
     DATA 1,20, 14,1, 16,48, [illegible] 11,20, 0,0
     DATA [illegible]SET STRIP, 14
     DATA [illegible] 0,0
     DATA [illegible]
     DATA [illegible]
     DATA [illegible]STAR BETA, 16
8320 DATA [illegible],40, 17,64, 15,1, 0,0
     DATA [illegible]
     DATA 16,64, 1,90, 19,88, 0,0
     DATA [illegible], 18
     DATA 1[illegible], 14,25, [illegible], 0,0
     DATA ULTIMA THULE, 1[illegible]
     DATA 17,80, 21,54, 0,0
     [illegible]
     [illegible] 1,80, 19,95, 0,0
8410 [illegible]WHERE, 0
8411 END
```

DIVISOR DE ÁGUAS

Estamos concluindo esta série de artigos sobre linguagem ASSEMBLY com um breve estudo da divisão binária e da programação das saídas na tela. No resumo do curso, revemos os principais tópicos abordados.

DIVISÃO DE 16 BITS POR 8 BITS					
Z80			6502		
START	LD	A,(DIVSR)	START	LDA	#$00
	LD	D,A		STA	QUOT
	LD	E,$00		STA	RMNDR
	LD	HL,(DVDND)		LDX	#$08
	LD	B,$08		LDA	DVDHI
LOOP0	AND	A		SEC	
	SBC	HL,DE		SBC	DIVSR
	INC	HL	LOOP0	PHP	
	JP	P,POSRES		ROL	QUOT
NEGRES	ADD	HL,DE		ASL	DVDLO
	DEC	HL		ROL	A
POSRES	ADD	HL,HL		PLP	
	DJNZ	LOOP0		BCC	CONT1
	LD	(QUOT),HL		SBC	DIVSR
	RET			JMP	CONT2
DIVSR	DB	$F9	CONT1	ADC	DIVSR
DVDND	DW	$FDE8	CONT2	DEX	
QUOT	DB	$00		BNE	LOOP0
RMNDR	DB	$00		BCS	EXIT
				ADC	DIVSR
				CLC	
			EXIT	ROL	QUOT
				STA	RMNDR
				RTS	
			DIVSR	DB	$F9
			DVDLO	DB	$E8
			DVDHI	DB	$FD
			QUOT	DB	$00
			RMNDR	DB	$00

Assim como usamos o método da multiplicação binária, também o método da divisão manual serve de modelo para a divisão binária. Considere esta divisão binária:

```
        00001110 r00        quociente
   1011)10011010            dividendo
       -1011                subtrair divisor
        10000
        -1011               subtrair divisor
          1011
         -1011              subtrair divisor
            00              não há resto
```

A essência desse método é a subtração repetida do divisor a partir dos bits superiores do dividendo. Dependendo do resultado dessa operação, transporta-se um 0 ou um 1 para o quociente. O resto é o resultado da última subtração de um divisor.

As várias maneiras de implementar esse algoritmo em linguagem ASSEMBLY não são tão evidentes como no caso da multiplicação. Mas também aqui a versão do Z80 usa a potência e a flexibilidade de seus registradores de 16 bits, enquanto o 6502 deve tomar e transportar 8 bits por vez. Estudaremos ambas as versões; o divisor fica no endereço representado pelo rótulo DIVSR, o dividendo em DVDND, o quociente em QUOT e o resto em RMNDR.

Observe que, em ambos os processadores, quando se subtrai o divisor do dividendo parcial, obtendo resultado negativo, deve-se restaurar o dividendo, somando-se outra vez o divisor. A versão do 6502 trabalha com o PSR (Processor Status Register, "registrador de estado do processador") após a subtração do divisor: há rotação do flag de transporte para o quociente, mas preserva-se também seu estado para indicar o resultado da subtração. Conseqüentemente, o PSR é introduzido na pilha antes da rotação e retirado logo depois, restaurando assim o transporte a seu estado pós-subtração.

Já examinamos, portanto, as quatro operações aritméticas, o que é importante como exercício de programação, pelo discernimento que traz sobre os processos da máquina; mas é desnecessário inventar combinações com a aritmética de 1 ou 2 bytes, já que há anos publicam-se tais rotinas em livros e revistas.

Saída na tela

Até agora usamos a RAM e a CPU para os cálculos, deixando os resultados de nossas operações em algum lugar da RAM para serem examinados manualmente por meio de um programa monitor. Esse método não é satisfatório; mas, sem conhecer a aritmética e as chamadas de sub-rotina, seria impossível obter saídas na tela com a linguagem ASSEMBLY.

A maioria dos micros tem tela de memória mapeada, ou seja, uma área da RAM dedicada a manter uma imagem na tela. A tela de vídeo compõe-se de pontos, ou pixels, que estão liga-

dos ou desligados, e que podem, portanto, ser representados por 1 ou 0 binários. Todo o conteúdo da tela é considerado como um "mapeamento", formado de pontos, dos bits que compõem a RAM de tela. Infelizmente, embora o Spectrum e o Commodore 64 usem essa técnica de mapeamento, não o fazem de forma direta. Para nossos propósitos, o método mais simples seria dividir cada linha da tela em pixels representados por bytes, numerados consecutivamente da esquerda para a direita, com o byte mais à esquerda numa linha seguindo o byte mais à direita da linha anterior. Por várias razões isso não ocorre em nenhuma dessas máquinas. Vamos considerar cada caso isoladamente.

A tela do Spectrum fica sempre no modo de alta resolução, e reserva-se uma determinada área da memória para seu mapeamento. Este, no entanto, é complexo, pois a tela se divide em três blocos horizontais de oito linhas de impressão, e cada linha divide-se, por sua vez, em oito linhas de pixels. O endereçamento dos bytes que as contêm se faz seqüencialmente dentro de cada linha, mas não entre elas. O Commodore 64 não segue esse padrão, mas tem igual complexidade. Por ora, será mais fácil entender o processo se nos limitarmos a produzir caracteres ASCII na tela.

Isso é algo que a máquina faz o tempo todo e, portanto, há, para esse fim, rotinas de código de máquina gravadas em ROM. Com uma descrição detalhada de sua operação, podemos chamar essas rotinas a partir de nossos programas em ASSEMBLY. O que precisamos saber é o endereço de chamada, os registradores de comunicações e algumas rotinas preparatórias.

No Spectrum não há rotinas preparatórias, e o registrador de comunicações é o acumulador, que deve conter o código ASCII do caractere a ser produzido. Basta apenas emitir a instrução RST $10 para que o caractere cujo código está no acumulador seja mostrado na tela, na posição atual do cursor. O código de operação RST (restart, "recomeçar") é um comando característico do Z80: trata-se de uma instrução de desvio de 1 byte, que opera na página zero, e utiliza um entre oito operandos possíveis — $00, $08,$10,$18 etc. até $38. Cada uma dessas posições indica o endereço inicial de uma rotina em ROM, situada em algum lugar da página zero.

Essas rotinas destinam-se em geral a processar entradas e saídas, e as chamamos por intermédio da instrução RST, em vez de fazê-lo diretamente pelo endereço. Isso garante maior velocidade (é mais rápido usar RST do que CALL, a nível de CPU) e também a compatibilidade do programa. Se todo programador de Z80 souber que RST $10 chama a rotina PRINT em qualquer máquina com Z80, ninguém vai precisar se preocupar em saber onde o analista de sistemas colocou a rotina PRINT, e ele ficará livre para colocá-la onde quiser, desde que a página zero seja estruturada de forma que as posições RST direcionem os programas para os endereços iniciais das rotinas.

No Commodore 64, o código ASCII no acumulador e o comando JSR $FFD2 geram o caractere na posição atual do cursor. Esta é a rotina CHKOUT, documentada no guia de referência do programador. Portanto, geralmente esse é o sistema usado de rotinas em ROM, e demonstra o princípio dos registradores de comunicações. A comunicação entre o programa principal e uma sub-rotina de entrada, por exemplo, pode passar certo caractere de um periférico para a CPU através do acumulador. Mesmo quando não há passagem de informações, um código de erro pode perfeitamente retornar da sub-rotina através de um dos registradores. Esse tipo de passagem está documentado nos muitos manuais de consulta hoje disponíveis, dedicados a equipamentos específicos. Concluímos este curso com um resumo de vários aspectos da programação em linguagem ASSEMBLY e em código de máquina.

Em resumo

Começamos o curso com uma visão geral do código de máquina, tentando desmitificá-lo e colocá-lo no contexto como apenas uma das linguagens utilizadas por programadores (e computadores). Vimos como a mesma seqüência de bytes na RAM é interpretada ora como um string de dados ASCII, ora como uma linha de um programa em BASIC, ora como uma sucessão de endereços de 2 bytes, e de novo como uma seqüência de comandos em código de máquina. Alguns minutos de experiências com um programa monitor em código de máquina bastam para convencer de que certas seqüências de bytes podem ser consideradas como três seqüências de instruções totalmente diferentes, mas válidas — dependendo de se começar a separação pelo primeiro, segundo ou terceiro byte. Nada instrínseco ao código impede que isso ocorra e, para a CPU, é indiferente executar um programa que o usuário escreveu ou alguma versão deturpada dele acidentalmente transportada para a memória.

Consideramos então a organização da memória e as convenções comuns de endereçamento. Para compreendê-las, estudamos a aritmética binária, o que de imediato delimitou nossa visão da CPU — nos processadores de 8 bits, temos de nos restringir, exceto em circunstâncias especiais, ao alcance de um só byte (ou seja, a faixa dos números decimais entre 0 e 255). Em vista da adequação da aritmética binária, evidenciaram-se as limitações do sistema decimal

no trabalho com a linguagem ASSEMBLY. Explorando a idéia da memória paginada, vimos como o tamanho das páginas lógicas deve ser função da base numérica e, num sistema binário, isso significa que o tamanho da página é uma potência de 2. Dois elevado à oitava dá 256 — o número-chave num sistema baseado em microprocessadores de 8 bits.

Como a aritmética binária logo se tornou demasiado incômoda e propensa a erros para ser usada como sistema de numeração, passamos para a aritmética hexadecimal (de base numérica 16). Vimos como é possível representar um byte de 8 bits por dois dígitos hexadecimais, de $00 a $FF, um dígito representando o estado dos quatro bits inferiores e o outro, o dos quatro superiores. Vimos com bastante detalhe a maneira como se armazenam os programas em BASIC na área reservada a programas. Ao estudarmos a simbolização como outra forma de código de máquina, fizemos uma incursão no sistema operacional. Descrevendo os marcadores de fim de linha, vimos como o interpretador BASIC reconhece onde termina um trecho de programa e outro começa. Também introduzimos a convenção de endereço baixo-alto, bem como o conceito de endereçamento indireto.

Daí passamos diretamente para a linguagem ASSEMBLY. Partimos de operações primitivas da CPU, comandadas pelos códigos de operação de 8 bits que constituem suas instruções de programação. Vimos em seguida os códigos mnemônicos do ASSEMBLY. Uma vez alcançado esse estágio, ficou claro que programar em código de máquina (ASSEMBLY) ou BASIC é tão somente programar, importando apenas resolver o problema lógico antes de nos preocuparmos com a codificação da solução. E solucionar problemas foi o tema central do curso; mas a falta de clareza de alguns conceitos do ASSEMBLY desviaram nossa atenção no sentido de eliminar a confusão que atrapalha a maioria das pessoas em seu primeiro contato com as linguagens computacionais de baixo nível.

O curso prosseguiu ensinando como carregar e executar programas em código de máquina em computadores que normalmente usam o BASIC. Examinamos as variáveis e os indicadores do sistema operacional, aprendendo como "roubar" espaço do BASIC. Passamos brevemente pela arquitetura dos microprocessadores Z80 e 6502 e começamos a escrever programas em ASSEMBLY que manipulavam a memória e o acumulador. Nesse ponto introduzimos os pseudocódigos de operação, que nos aproximaram da maneira real de programar e nos afastaram do código de máquina, da montagem manual e do trabalhoso detalhamento da programação de baixo nível.

Devido à necessidade de estruturas lógicas, passamos a considerar o PSR (Processor Status Register, "registrador de estado do processador"). Sua função de registrar os resultados das operações da CPU foi apresentada com uma introdução à aritmética binária, usando a instrução ADC ("soma com transporte"). A importância do PSR e do flag de transporte ficou esclarecida. Daí em diante, o curso concentrou-se no PSR e nas instruções a ele associadas.

Examinamos brevemente os vários modos de endereçamento: deu-se mais atenção ao indexado, devido à sua importância na manipulação de loops, listas e tabelas. Logo se evidenciou a necessidade de um tipo de instrução para alterar o fluxo de controle de um programa; e começamos a examinar as instruções de desvio condicional, ainda explorando o potencial dos modos de endereçamento direto e indireto. Com os desvios condicionais, aritmética básica e estruturas do tipo matriz, obtivemos o esqueleto de qualquer linguagem de programação, restando-nos preenchê-lo com a prática e a investigação sistemática. Examinamos o método de chamada e retorno de sub-rotinas, também como uma forma de apresentar a última área inexplorada do sistema operacional — a pilha. Vendo como ela funciona, para que serve e como a usamos, acrescentamos alguns novos recursos ao nosso repertório de programação em código de máquina. Um exame mais detalhado dos registradores da CPU e de suas interações apresentou novas possibilidades de manipulação da memória e do processador.

Por fim, com um conhecimento operacional da arquitetura do microprocessador e um vocabulário de códigos de operação, abordamos a aritmética binária, incluindo as singularidades da subtração e do complemento de 2, bem como as complexidades da multiplicação e da divisão.

Respostas dos exercícios anteriores

A) A solução de execução mais rápida é uma rotina escrita especificamente para multiplicandos de 16 bits, semelhantes à rotina para 8 bits publicada no último artigo. Por outro lado, dividindo a multiplicação de 16 bits em duas multiplicações de 8 bits (multiplicador) pelo byte inferior, depois pelo superior, podemos chamar duas vezes a rotina de 8 bits, ajustar o transporte do byte inferior e armazenar o resultado nos bytes do produto.

B) Uma rotina de multiplicação que usa a adição repetida consiste simplesmente num loop cujo contador é o valor do multiplicador; cada vez que se executa o loop, soma-se o multiplicando ao produto.

ROR
ROTAÇÃO PARA A DIREITA
Registrador — 6A (1 byte)
6502
O conteúdo do byte fez rotação de 1 bit para a direita através do transporte.
EFEITO SOBRE O PSR
S V B D I Z C
MSB X X X LSB
Exemplo:
LOCALIZAÇÃO CÓDIGO DE MÁQUINA LINGUAGEM ASSEMBLY
C100 6A ROR A
ANTES DEPOIS
PSR ???????1 1??????01
A 01011001 10101100
6A $C100
Memória de dados
Memória de programa

RR
ROTAÇÃO PARA A DIREITA
Registrador CB (2 bytes)
Z80
O conteúdo do byte fez rotação de 1 bit para a direita através do transporte.
EFEITO SOBRE O PSR
S Z H V N C
MSB X X 0 X 0 X LSB
Exemplo:
LOCALIZAÇÃO CÓDIGO DE MÁQUINA LINGUAGEM ASSEMBLY
C100 CB 1C RR H
ANTES DEPOIS
PSR ???????1 10?0?001
A 01011001 10101100
CB $C100
23 $C101
Memória de dados
Memória de programa

SBC
SUBTRAIR COM EMPRÉSTIMO
Absoluto - ED (3 bytes)
6502
Subtrai-se do acumulador o conteúdo da posição da memória; o transporte mostra empréstimo.
EFEITO SOBRE O PSR
S V B D I Z C
MSB X X X X LSB
Exemplo:
LOCALIZAÇÃO CÓDIGO DE MÁQUINA LINGUAGEM ASSEMBLY
C100 ED A3 59 SBC $59A3
ANTES DEPOIS
PSR ???????1 00?????01
A A7 22
$59A3 85
ED $C100
A3 $C101
59 $C102
Memória de dados
Memória de programa

SBC
SUBTRAIR COM EMPRÉSTIMO
Imediato — DE (2 bytes)
Z80
Subtrai-se do acumulador o conteúdo do byte que se segue aos códigos de operação. accumulator.
EFEITO SOBRE O PSR
S Z H V N C
MSB X X X X X LSB
Exemplo:
LOCALIZAÇÃO CÓDIGO DE MÁQUINA LINGUAGEM ASSEMBLY
C100 DE 85 SBC A,$85
ANTES DEPOIS
PSR ???????0 00?0?000
A A7 22
DE $C100
85 $C101
Memória de dados
Memória de programa

ASL
DESLOCAMENTO ARITMÉTICO PARA A ESQUERDA
Registrador — 0A (1 byte)
6502
O conteúdo do byte desloca-se 1 bit para a esquerda através do transporte; entra 0 no bit LSB.
EFEITO SOBRE O PSR
S V B D I Z C
MSB X X X LSB
Exemplo:
LOCALIZAÇÃO CÓDIGO DE MÁQUINA LINGUAGEM ASSEMBLY
C100 0A ASL A
ANTES DEPOIS
PSR ???????? 0?????11
A 10000000 00000000
0A $C100
Memória de dados
Memória de programa

SLA
DESLOCAMENTO ARITMÉTICO PARA A ESQUERDA
Registrador — CB (2 bytes)
Z80
O conteúdo do byte desloca-se 1 bit para a esquerda através do transporte; entra 0 no bit LSB.
EFEITO SOBRE O PSR
S Z H V N C
MSB X X 0 X 0 X LSB
Exemplo:
LOCALIZAÇÃO CÓDIGO DE MÁQUINA LINGUAGEM ASSEMBLY
C100 CB 23 SLA E
ANTES DEPOIS
PSR ???????? 01?0?001
E 10000000 00000000
0
CB $C100
23 $C101
Memória de dados
Memória de programa

CMP
COMPARAR O ACUMULADOR
Absoluto - CD (3 bytes)
6502
Compara-se o acumulador com o conteúdo da posição da memória.
EFEITO SOBRE O PSR
S V B D I Z C
MSB X X X LSB
Exemplo:
LOCALIZAÇÃO CÓDIGO DE MÁQUINA LINGUAGEM ASSEMBLY
C100 CD 7E 40 CMP $407E
ANTES DEPOIS
PSR ???????? 0?????10
A 7D 7D
$407E 7D
CD $C100
7E $C101
40 $C102
Memória de dados
Memória de programa

CP
COMPARAR O ACUMULADOR
Imediato — FE (2 bytes)
Z80
Compara-se o acumulador com o conteúdo do byte que se segue ao código de operação. accumulator.
EFEITO SOBRE O PSR
S Z H V N C
MSB X X X X 1 X LSB
Exemplo:
LOCALIZAÇÃO CÓDIGO DE MÁQUINA LINGUAGEM ASSEMBLY
C100 FE 7D CP $7D
ANTES DEPOIS
PSR ???????? 01?0?010
A 7D 7D
FE $C100
7D $C101
Memória de dados
Memória de programa

VISÃO DO FUTURO

Não conhecemos ainda o efeito das máquinas dotadas de inteligência criativa sobre o homem. Mas é certo que terá um impacto decisivo sobre as gerações futuras — de seres humanos e de computadores.

O que nos reserva o futuro
Os progressos da biotecnologia poderão nos abrir a perturbadora perspectiva dos biônicos, híbridos homem-máquina, combinando os processos intuitivos do cérebro humano com o potencial lógico e matemático dos microprocessadores. Se for esse o caso, assistiremos à evolução humana sair das mãos da natureza e ficar sob o controle de homens-máquina capazes de auto-reprodução seletiva.

Comparando o desenvolvimento em computação a uma escada, podemos, num sentido amplo, distinguir quatro degraus, de crescente complexidade. No primeiro, temos as tarefas corriqueiras do guarda-livros, que os computadores desempenham muito bem. Um degrau acima, estão os programas que ajudam as pessoas a tomarem decisões inteligentes — sistemas de revisão financeira e planilhas, por exemplo. Essas ferramentas precisam ser flexíveis, uma vez que não se pode conhecer de antemão as exigências do usuário.

Ao nível do terceiro degrau, encontramos a aplicação de habilidades derivadas daquelas dos seres humanos. É aí que se concentram as aplicações da IA (Inteligência Artificial), agora e no futuro imediato. Um software como esse tem exemplo no sistema de habilidade Prospector, que codificou os procedimentos de vários geólogos. A aplicação prática dessas normas de trabalho à prospecção culminou com a descoberta de grandes depósitos de molibdênio no Estado de Washington, até então desconhecidos. Depois disso, outro depósito foi descoberto no Canadá.

As façanhas do Prospector passaram para a mitologia da IA, e de fato impressionam. Mas não nos levam ao quarto degrau da escada: o das máquinas dotadas de inteligência criativa. Para compreender isso, precisamos fazer uma contraposição entre produtividade e criatividade.

A produtividade depende da obediência a regras — e nisso os computadores são excelentes. Eles podem se mostrar bem mais produtivos que as pessoas, mas daí a torná-los criativos vai grande distância, constituindo tarefa extremamente difícil. Para atingirem o atributo de criativos, os programas de computador precisarão ser capazes de elaborar suas próprias regras. E, de fato, alguns pesquisadores que trabalham nas fronteiras da IA estão tentando conseguir nada menos que isso.

Um programa que deu um passo na direção do quarto degrau é o Eurisko, projetado por Doug Lenat, da Universidade de Stanford. Trata-se de um programa de descoberta que tem sido aplicado em vários domínios, desde jogos de batalha naval até desenhos de circuitos VLSI (Very Large Scale Integration, "integração em escala muito ampliada"). O Eurisko parte de uma coleção de regras e conceitos heurísticos e os aplica aos domínios escolhidos. Muita coisa no programa é padronizada, consistindo a novidade no fato de ele poder modificar sua própria heurística. Isso dá ao sistema um grande poder, visto ser ele capaz de adaptar e especializar suas regras genéricas, a fim de lidar com situações novas.

O Eurisko fez uma descoberta que mais tarde resultou numa patente, e isso dificilmente pode ser considerado trivial. Trata-se da descoberta de um novo desenho para o circuito lógico em 3 D (uma porta NÃO-E/OU), sobre o qual jamais alguém no Vale do Silício — ou em qualquer outro lugar — havia pensado. No entanto, originou-se da aplicação de uma única regra.

Quando o Eurisko foi aplicado ao projeto do VLSI, ele já possuía uma regra heurística, proveniente de tarefas anteriores, que afirmava o seguinte: "Se um conceito é interessante, tente torná-lo ainda mais". Aplicada a um dispositivo em 2 D, tal regra leva a uma versão mais simétrica, em 3 D, desse dispositivo — recentemente fabricado com sucesso. Um componente em 3 D, depois de isolados os problemas de manufatura, pode ser compactado mais densamente, oferecendo maior capacidade.

A criatividade é vista como o ponto máximo da inteligência humana, e encontra-se mergulhada em nebulosas explicações envolvendo intuição e lampejos; mas o Eurisko pode nos dar a necessária pausa para pensar. Constitui exemplo de autêntica descoberta resultante da aplicação

Da máquina ao super-homem
HAL, o computador do filme *2001: Uma odisseia no espaço*, visto aqui numa cena de sua continuação, *2010, o ano em que faremos contato*, começou sua existência com capacidade para sintetizar a fala e dotado de um complexo conjunto de regras, pelas quais podia avaliar o sucesso de uma missão. Tais características são hoje implementadas em proporção cada vez maior no hardware e no software. No livro *2010*, Arthur C. Clarke levou o conceito de inteligência da maquina varios passos adiante, fazendo HAL transcender seu status mecânico e se tornar uma figura super-humana, questionando o pressuposto de que as máquinas não podem evoluir independentemente de seus criadores

de uma só regra. O computador criativo encontra-se mais perto de nós do que pensa a maioria das pessoas.

O lado obscuro da IA

Até aqui, esta série de artigos concentrou-se na face aceitável da IA, mas existe aquilo que alguns consideram o lado obscuro da IA. Independentemente de quanto sejam impressionantes os progressos na pesquisa da IA, podemos não gostar de algumas de suas aplicações — especialmente se levarmos em conta que boa parte do trabalho de pesquisa da IA depende da ajuda financeira militar. Mais de metade de suas aplicações a curto prazo visa o campo de batalha. Essas aplicações incluem:

- Submersíveis inteligentes;
- Munições "perspicazes";
- Misseis de cruzeiros;
- Tanques autodirigidos;
- Sistemas de sonar baseados em conhecimentos;
- Torpedos que procuram o alvo;
- Sistemas de produção de imagens de radar

e muitos outros dispositivos dos quais poucos de nós já ouviram falar.

Mais alarmante, porém, é o que algumas pessoas imaginam como resultado de pesquisas excessivamente bem-sucedidas com vistas à criação de MUIs (Máquinas Ultra-Inteligentes). Essas pessoas vislumbram um futuro dominado pelas MUIs. Incapazes de realmente "pensar", as máquinas poderão, no entanto, fazer — e bem melhor que o ser humano — quase tudo o que exige atitude racional. Na matemática e nas ciências naturais, irão muito além de nós. Na indústria, a persistir o atual panorama, elas excederão a tal ponto os administradores humanos que a execução de toda a economia estará sob seu controle. Em resumo: seremos intelectualmente inferiores a elas.

A idéia de seres "biônicos" — híbridos, meio homens meio máquinas — tem sido uma das favoritas entre os escritores de ficção científica. Mas recentemente ela se transferiu do campo da imaginação para a área da conjectura científica a longo prazo. A ascendência de duas áreas da pesquisa científica aparentemente incompatíveis — a engenharia genética e a engenharia do conhecimento — é em grande parte responsável por isso.

A engenharia genética está interessada na manipulação do código genético na matéria viva, de modo a "programar" melhorias genéticas em gerações sucessivas. Os engenheiros geneticistas já aperfeiçoaram técnicas que lhes permitem enxertar características desejáveis em microorganismos. Com isso, conseguem-se drogas que antes atingiam custos proibitivos ou eram simplesmente impossíveis de fabricar em larga escala. Espera-se que, num futuro próximo, a engenharia genética desenvolva os chamados biochips. Programando a ação das enzimas que dividem e reconstituem moléculas, será possível fazer os circuitos lógicos *crescerem* a partir de moléculas de proteínas, o que resultará numa formidável redução de tamanho. A idéia de utilizar material vivo para transportar mensagens elétricas codificadas não é nova; o sistema nervoso funciona assim.

Atualmente já se consegue alterar a composição genética da carne para torná-la mais resistente ao ataque de fungos. O próximo passo será utilizar essa técnica para erradicar doenças genéticas dos seres humanos, como a síndrome de Down, por exemplo. Assim que forem dados os primeiros passos na manipulação dos genes humanos, eventualmente surgirão outros experimentos — digamos, os cientistas poderão tentar enxertar no cérebro humano um bio-ROM contendo um banco de dados. Os cientistas já obtiveram algum sucesso em enxertar dispositivos elétricos no cérebro. Uma estudante surda de Oxford recebeu em seu cérebro, nas áreas responsáveis pela audição, a saída elétrica de um microfone. Após breve período de treinamento, ela conseguiu extrair sentido dos sinais recebidos — o suficiente para "ouvir" sons. Em poucas palavras: parece que o cérebro é capaz de aprender a se colocar como interface em relação a fontes elétricas externas.

Algumas pessoas propõem que, num futuro distante, nossos descendentes evolutivos possam ser algum tipo de híbrido homem-máquina altamente avançado. Como o processo evolutivo parece fazer uso de qualquer material à mão — os pulmões, por exemplo, mudam em órgão de flutuação —, parece razoável supor que esses organismos híbridos venham a abrigar, em seus corações, um cérebro humano circundado por várias camadas de tecnologia avançada. De fato, em sua forma atual, o cérebro humano apresenta várias camadas, que se desenvolveram durante a nossa evolução.

As aplicações dessas duas novas tecnologias são ainda muito incertas. E a realização ou não das hipóteses aqui esboçadas parece depender mais do enfoque social (constituem algo desejável?) do que da praticabilidade técnica.

SUMÁRIO

VOLUME 2

APLICAÇÕES

CHIPS & BYTES

CIÊNCIA DA COMPUTAÇÃO

LINGUAGENS

PERFIL

PERIFÉRICOS

PROJETOS DE PROGRAMAS

RAIO X

SOFTWARE

ÍNDICE

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z